Lattice Reduction for Modular Knapsack*

Thomas Plantard, Willy Susilo, and Zhenfei Zhang

Centre for Computer and Information Security Research
School of Computer Science & Software Engineering (SCSSE)
University of Wollongong, Australia
{thomaspl,wsusilo,zz920}@uow.edu.au

Abstract. In this paper, we present a new methodology to adapt any kind of lattice reduction algorithms to deal with the modular knapsack problem. In general, the modular knapsack problem can be solved using a lattice reduction algorithm, when its density is low. The complexity of lattice reduction algorithms to solve those problems is upper-bounded in the function of the lattice dimension and the maximum norm of the input basis. In the case of a low density modular knapsack-type basis, the weight of maximum norm is mainly from its first column. Therefore, by distributing the weight into multiple columns, we are able to reduce the maximum norm of the input basis. Consequently, the upper bound of the time complexity is reduced.

To show the advantage of our methodology, we apply our idea over the floating-point LLL (L^2) algorithm. We bring the complexity from $O(d^{3+\varepsilon}\beta^2 + d^{4+\varepsilon}\beta)$ to $O(d^{2+\varepsilon}\beta^2 + d^{4+\varepsilon}\beta)$ for $\varepsilon < 1$ for the low density knapsack problem, assuming a uniform distribution, where d is the dimension of the lattice, β is the bit length of the maximum norm of knapsack-type basis.

We also provide some techniques when dealing with a principal ideal lattice basis, which can be seen as a special case of a low density modular knapsack-type basis.

Keywords: Lattice Theory, Lattice Reduction, Knapsack Problem, LLL, Recursive Reduction, Ideal Lattice.

1 Introduction

To find the shortest non-zero vector within an arbitrary lattice is an NP-hard problem [1]. Moreover, till now there is no polynomial algorithm that finds a vector in the lattice that is polynomially close to the shortest non-zero vector. However, there exist several algorithms, for example, LLL [14] and L^2 [19], running in polynomial time in d and β, where d is the dimension of the lattice, and β is the bit length of the maximum norm of input basis, that find vectors with exponential approximation (in d) to the shortest non-zero vector. Indeed, in some lattice based cryptography/cryptanalysis, it may not be necessary to recover the exact shortest non-zero vector, nor a polynomially close one. Finding one with

* This work is supported by ARC Future Fellowship FT0991397.

L.R. Knudsen and H. Wu (Eds.): SAC 2012, LNCS 7707, pp. 275–286, 2013.

exponential distance to the shortest one is already useful, for instance, to solve a low density knapsack problem or a low density modular knapsack problem.

Definition 1 (Knapsack Problem). *Let $\{X_1, X_2, \ldots, X_d\}$ be a set of positive integers. Let $c = \sum_1^d s_i X_i$, where $s_i \in \{0, 1\}$. A knapsack problem is given $\{X_i\}$ and c, find each s_i.*

The density of a knapsack, denoted by ρ, is d/β, where β is the maximum bit length of X_i-s.

Definition 2 (Modular Knapsack Problem). *Let $\{X_0, X_1, \ldots, X_d\}$ be a set of positive integers. Let $c = \sum_1^d s_i X_i \bmod X_0$, where $s_i \in \{0, 1\}$. A modular knapsack problem is given $\{X_i\}$ and c, find each s_i.*

The knapsack problem is also known as the subset sum problem [12]. When $\sum s_i \ll d$, it becomes a sparse subset sum problem (SSSP). The decisional version of the knapsack problem is NP-complete [9]. However, if its density is too low, there is an efficient reduction to the problem of finding the shortest vector from a lattice (refer to [13,21,3]).

In this paper, we deal with a (modular) knapsack problem assuming a uniform distribution, i.e., X_i-s are uniformly randomly distributed.

We refer to B_K as the *knapsack-type basis*, and B_M as the *modular knapsack-type basis*. In the rest of the paper, for simplicity, we focus on knapsack-type basis, although the adoption over a modular knapsack-type basis is straightforward.

$$B_K = \begin{pmatrix} X_1 & 1 & 0 & \ldots & 0 \\ X_2 & 0 & 1 & \ldots & 0 \\ X_3 & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ X_d & 0 & 0 & \ldots & 1 \end{pmatrix}, \qquad B_M = \begin{pmatrix} X_0 & 0 & 0 & \ldots & 0 \\ X_1 & 1 & 0 & \ldots & 0 \\ X_2 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ X_d & 0 & 0 & \ldots & 1 \end{pmatrix}.$$

We also consider a *principal ideal lattice basis*. A principal ideal lattice is an ideal lattice that can be represented by two integers. This type of lattice enables important applications, for instance, constructing fully homomorphic encryption schemes with smaller key size (see [5,26] for an example of this optimization).

A basis of a principal ideal lattice (see Section 5) maintains a similar form of modular knapsack basis, with $X_i = -\alpha^i \bmod X_0$ for $i \geq 1$, where α is the root for the principal ideal. The security of the corresponding cryptosystem is based on the complexity of reducing this basis.

In general, to solve any of the above problems using lattice, one always start by performing an LLL reduction on a lattice $\mathcal{L}(B_K)$ ($\mathcal{L}(B_M)$, respectively). Then depending on the type of problem and the result the LLL algorithm produces, one may perform stronger lattice reduction (BKZ [24,7,2] for example) and/or use enumeration techniques such as Kannan SVP solver [8].

To date, the complexity of best LLL reduction algorithms for the above three type of basis is upper bounded by $O(d^{3+\varepsilon}\beta^2 + d^{4+\varepsilon}\beta)$ [19], although heuristically, one is able to obtain $O(d^{2+\varepsilon}\beta^2)$ in practice when ρ is significantly smaller than 1 [20].

Our Contribution: We propose a new methodology to reduce low density modular knapsack-type basis using LLL-type algorithms. Instead of reducing the whole knapsack-type basis directly, we pre-process its sub-lattices, and therefore, the weight of X_i-s is equally distributed into several columns and the reduction complexity is thereafter reduced. Although the idea is somewhat straightforward, the improvement is very significant.

Table 1. Comparison of time complexity

Algorithms	Time Complexity
LLL[14]	$O(d^{5+\varepsilon}\beta^{2+\varepsilon})$
LLL for knapsack	$O(d^{4+\varepsilon}\beta^{2+\varepsilon})$
L^2[19]	$O(d^{4+\varepsilon}\beta^2 + d^{5+\varepsilon}\beta)$
L^2 for knapsack[19]	$O(d^{3+\varepsilon}\beta^2 + d^{4+\varepsilon}\beta)$
$\tilde{\mathrm{L}}^1$[22]	$O(d^{\omega+1+\varepsilon}\beta^{1+\varepsilon} + d^{5+\varepsilon}\beta)$
Our rec-L^2	$O(d^{2+\varepsilon}\beta^2 + d^{4+\varepsilon}\beta)$
Our rec-L^1	$O(d^{\omega}\beta^{1+\varepsilon} + d^{4+\varepsilon}\beta)$

Table 1 shows a time complexity comparison between our algorithms and the existing algorithms. However, we note that the complexities of all the existing algorithms are in worst-case, or in another words, for any type of basis, the algorithms will terminate in the corresponding time. In contrast, in our algorithm, we assume a uniform distribution, and therefore, the complexity given for our recursive reduction algorithms is the upper bound following this assumption. Nevertheless, we note that such an assumption is quite natural in practice.

Our result is also applicable to a principal ideal lattice basis. In addition, we provide a technique that further reduces the time complexity. Note that our technique does not affect the asymptotic complexity as displayed in Table 1.

Paper Organization: In the next section, we review some related area to this research. In Section 3, we propose our methodology to deal with low density knapsack-type basis, introduce our recursive reduction algorithm, and analyze its complexity. Then, we apply our method to L^2 and compare its complexity with non-modified L^2 in Section 4. In Section 5, we analyze the special case of the principal ideal lattice basis. Finally, the last section concludes this paper.

2 Background

2.1 Lattice Basics

In this subsection, we review some concepts of the lattice theory that will be used throughout this paper. The lattice theory, also known as the geometry of numbers, was introduced by Minkowski in 1896 [17]. We refer readers to [15,16] for a more complex account.

Definition 3 (Lattice). *A lattice $\mathcal{L}$ is a discrete sub-group of $\mathbb{R}^n$, or equivalently the set of all the integral combinations of $d \leq n$ linearly independent vectors over $\mathbb{R}$.*

$$\mathcal{L} = \mathbb{Z}b_1 + \mathbb{Z}b_2 + \cdots + \mathbb{Z}b_d, b_i \in \mathbb{R}^n$$

$B = (b_1, \ldots, b_d)$ is called a basis of $\mathcal{L}$ and d is the dimension of $\mathcal{L}$, denoted as $\dim(\mathcal{L})$. $\mathcal{L}$ is a full rank lattice if d equals to n.

For a given lattice $\mathcal{L}$, there exists an infinite number of basis. However, its determinant (see Definition 4) is unique.

Definition 4 (Determinant). *Let $\mathcal{L}$ be a lattice. Its determinant, denoted as $\det(\mathcal{L})$, is a real value, such that for any basis B of $\mathcal{L}$, $\det(\mathcal{L}) = \sqrt{\det(B \cdot B^T)}$, where B^T is the transpose of B.*

Definition 5 (Successive Minima). *Let $\mathcal{L}$ be an integer lattice of dimension d. Let i be a positive integer. The i-th successive minima with respect to $\mathcal{L}$, denoted by λ_i, is the smallest real number, such that there exist i non-zero linear independent vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_i \in \mathcal{L}$ with*

$$\|v_1\|, \|v_2\|, \ldots, \|\boldsymbol{v}_i\| \leq \lambda_i.$$

The i-th minima of a random lattice (as defined in Theorem 1) is estimated by:

$$\lambda_i(\mathcal{L}) \sim \sqrt{\frac{d}{2\pi e}} \det(\mathcal{L})^{\frac{1}{d}}. \tag{1}$$

Definition 6 (Hermite Factor). *Let $B = (b_1, \ldots, b_d)$ a basis of $\mathcal{L}$. The Hermite factor with respect to B, denoted by $\gamma(B)$, is defined as $\frac{\|b_1\|}{\det(\mathcal{L})^{\frac{1}{d}}}$.*

Note that Hermite factor indicates the quality of a reduced basis.
Additionally, following the result of [6]:

Theorem 1 (Random Lattice). *Let B be a modular knapsack-type basis constructing from a modular knapsack problem given by $\{X_i\}$. $\mathcal{L}(B)$ is a random lattice, if $\{X_i\}$ are uniformly distributed.*

2.2 Lattice Reduction Algorithms

In 1982, Lenstra, Lenstra and Lovasz [14] proposed an algorithm, known as LLL, that produces an LLL-reduced basis for a given basis. For a lattice $\mathcal{L}$ with dimension d, and a basis B, where the norm of all spanning vectors in B is $\leq 2^\beta$, the worst-case time complexity is polynomial $O(d^{5+\varepsilon}\beta^{2+\varepsilon})$. Moreover, it is observed in [20] that in practice, LLL seems to be much more efficient in terms of average time complexity.

In 2005, Nguyen and Stehlé [19] proposed an improvement of LLL, which is the first variant whose worst-case time complexity is quadratic with respect to β.

This algorithm is therefore named L^2. This algorithm makes use of floating-point arithmetics, hence, the library that implements L^2 is sometimes referred to as *fplll* [23]. It terminates with a worst-case time complexity of $O(d^{4+\varepsilon}\beta^2 + d^{5+\varepsilon}\beta)$ for any basis. For a knapsack-type basis, it is proved that L^2 terminates in $O(d^{3+\varepsilon}\beta^2 + d^{4+\varepsilon}\beta)$, since there are $O(d\beta)$ loop iterations for these bases instead of $O(d^2\beta)$ for random bases (see Remark 3, [19]). Moreover, some heuristic results show that when dealing with this kind of bases, and when d, β grow to infinity, one obtains $\Theta(d^3\beta^2)$ when $\beta = \Omega(d^2)$, and $\Theta(d^2\beta^2)$ when β is significantly larger than d (see Heuristic 3, [20]).

Recently, in 2011, Novocin, Stehlé and Villard [22] proposed a new improved LLL-type algorithm that is quasi-linear in β. This led to the name $\tilde{\mathrm{L}}^1$. It is guaranteed to terminate in time $O(d^{5+\varepsilon}\beta + d^{\omega+1+\varepsilon}\beta^{1+\varepsilon})$ for any basis, where ω is a valid exponent from matrix multiplications. To bound ω, we have $2 < \omega \leq 3$. A typical setting in [22] is $\omega = 2.3$.

In [27], van Hoeij and Novocin proposed a gradual sub-lattice reductions algorithm based on LLL that deals with knapsack-type basis. Unlike other LLL-type reduction algorithms, it only produces a basis of a sub-lattice. This algorithm uses a worst-case $O(d^7 + d^3\beta^2)$ time complexity.

For more improvements on LLL with respect to d, we refer readers to [18,25,10,11].

With regard to the quality of a reduced basis for an arbitrary lattice, the following theorem provides an upper bound.

Theorem 2. *For a lattice $\mathcal{L}$, if $(b_1, \ldots, b_n)$ form an LLL-reduced basis of $\mathcal{L}$, then,*

$$\forall i, \|b_i\| \leq 2^{\frac{d-1}{2}} \max(\lambda_i(\mathcal{L})). \tag{2}$$

Therefore, assuming a uniform distribution, we have the following.

1. If a modular knapsack problem follows a uniform distribution, then its corresponding basis forms a random lattice.
2. If $\mathcal{L}$ is a random lattice, then $\lambda_i(\mathcal{L})$ is with respect to Equation 1.
3. From Equation 1 and 2, we have $\|b_i\| \leq 2^{\frac{d-1}{2}} \sqrt{\frac{d}{2\pi e}} \det(\mathcal{L})^{\frac{1}{d}}$ for a random lattice.

Hence, for a modular knapsack-type basis, if $B = (b_1, \ldots, b_d)$ forms its LLL-reduced basis, then

$$\|b_i\| < 2^d \det(\mathcal{L})^{\frac{1}{d}}, \quad 1 \leq i \leq d.$$

In terms of the quality of $\|b_1\|$, the work in [4] shows that on average cases, LLL-type reduction algorithms is able to find a short vector with a Hermite factor 1.0219^d, while on worst cases, 1.0754^d, respectively. Further, heuristically, it is impossible to find vectors with Hermite factor $< 1.01^d$ using LLL-type algorithms [4]. By contrast, a recent work of BKZ 2.0 [2] finds a vector with a Hermite factor as small as 1.0099^d.

It has been shown that other lattice reduction algorithms, for instance, BKZ [24,7], and BKZ 2.0 [2], produce a basis with better quality. However, in general,

they are too expensive to use. We also note those methods require to perform at least one LLL reduction.

As for low density knapsack-type basis, the Hermite factor of the basis is large. This implies that, in general, the output basis of most reduction algorithms contains the demanded short vector. In this case, the time complexity is more important, compared with the quality of the reduced basis/vectors. For this reason, in this paper, we focus only on LLL-type reduction algorithms.

3 Our Reduction Methodology

3.1 A Methodology for Lattice Reduction

In this subsection, we do not propose an algorithm for lattice reduction but rather a methodology applicable to *all* lattice reduction algorithms for the knapsack problem with uniform distribution.

Let $\mathcal{A}$ be an LLL-type reduction algorithm that returns an LLL-reduced basis B_{red} of a lattice $\mathcal{L}$ of dimension d, where $B_{red} = (b_1, \ldots, b_d)$, $0 < \|b_i\| < c_0^d \det(\mathcal{L})^{\frac{1}{d}}$ for certain constant c_0. The running time will be $c_1 d^{a_1}\beta^{b_1} + c_2 d^{a_2}\beta^{b_2}$, where a_1, b_1, a_2 and b_2 are all parameters, c_1 and c_2 are two constants. Without losing generality, assuming $a_1 \geq a_2$, $b_1 \leq b_2$ (if not, then one term will overwhelm the other, and hence, making the other term negligible). We note that this is a formalization of all LLL-type reduction algorithms.

For a knapsack-type basis B of $\mathcal{L}$, where most of the weight of β are from the first column of the basis matrix $B = (b_1, b_2, \ldots, b_d)$, it holds that $2^\beta \sim \det(\mathcal{L})$. Moreover, for any sub-lattice $\mathcal{L}_s$ of $\mathcal{L}$ that is spanned by a subset of row vectors $\{b_1, b_2, \ldots, b_d\}$, it is easy to prove that $\det(\mathcal{L}_s) \sim 2^\beta$. In addition, since we assume a uniform distribution, the sub-lattice spanned by the subset of vectors can be seen as a random lattice. Note that the bases of those sub-lattice are knapsack-type basis, so if one needs to ensure the randomness, one is required to add a new vector $\langle X_0, 0, \ldots, 0\rangle$ to the basis and convert it to a modular one. One can verify that this modification will not change the asymptotic complexity. Nevertheless, in practice, it is natural to omit this procedure.

We firstly pre-process the basis, so that the weight is as equally distributed into all columns as possible, and therefore, the maximum norm of the new basis is reduced. Suppose we cut the basis into d/k blocks and each block contains k vectors. Then one applies $\mathcal{A}$ on each block. Since we know that the determinant of each block is $\sim 2^\beta$, this pre-processing gives us a basis with smaller maximum norm $\sim c_0^k 2^{\beta/k}$. Further, since the pre-processed basis and the initial basis span the same lattice, the pre-processing will not affect the quality of reduced basis that a reduction algorithm returns.

Below, we show an example of how this methodology works with dimension 4 knapsack-type basis, where we cut $\mathcal{L}$ into two sub-lattices and pre-process them independently. As a result, $X_i \sim 2^\beta$, while $x_{i,j} \lesssim c_0^2 2^{\frac{\beta}{2}}$ for a classic LLL-type reduction algorithm.

$$B_{before} = \begin{pmatrix} X_1 & 1 & 0 & 0 & 0 \\ X_2 & 0 & 1 & 0 & 0 \\ \hline X_3 & 0 & 0 & 1 & 0 \\ X_4 & 0 & 0 & 0 & 1 \end{pmatrix} \Longrightarrow B_{after} = \begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} & 0 & 0 \\ x_{2,1} & x_{2,2} & x_{2,3} & 0 & 0 \\ \hline x_{3,1} & 0 & 0 & x_{3,4} & x_{3,5} \\ x_{4,1} & 0 & 0 & x_{4,4} & x_{4,5} \end{pmatrix}$$

Now we examine the complexity. The total time complexity of this pre-processing is $c_1 dk^{a_1-1}\beta^{b_1} + c_2 dk^{a_2-1}\beta^{b_2}$. The complexity of the final reduction now becomes $c_1 d^{a_1}(k\log_2(c_0)+\beta/k)^{b_1} + c_2 d^{a_2}(k\log_2(c_0)+\beta/k)^{b_2}$. Therefore, as long as

$$\begin{aligned} c_1 d^{a_1}(k\log_2(c_0)+\beta/k)^{b_1} + c_2 d^{a_2}(k\log_2(c_0)+\beta/k)^{b_2} \\ + c_1 dk^{a_1-1}\beta^{b_1} + c_2 dk^{a_2-1}\beta^{b_2} < c_1 d^{a_1}\beta^{b_1} + c_2 d^{a_2}\beta^{b_2}, \end{aligned} \tag{3}$$

conducting the pre-processing will reduce the complexity of whole reduction.

In the case where $k\log_2(c_0)$ is negligible compared with β/k, we obtain:

$$\begin{aligned} c_1 d^{a_1}(\beta/k)^{b_1} + c_2 d^{a_2}(\beta/k)^{b_2} + c_1 dk^{a_1-1}\beta^{b_1} + c_2 dk^{a_2-1}\beta^{b_2} \\ < c_1 d^{a_1}\beta^{b_1} + c_2 d^{a_2}\beta^{b_2}. \end{aligned}$$

Therefore,

$$c_1\left(d^{a_1} - \frac{d^{a_1}}{k^{b_1}} - dk^{a_1-1}\right)\beta^{b_1} + c_2\left(d^{a_2} - \frac{d^{a_2}}{k^{b_2}} - dk^{a_2-1}\right)\beta^{b_2} > 0.$$

Taking L^2 as an example, where $a_1 = 4$, $b_1 = 2$, $a_2 = 5$ and $b_2 = 1$, let $k = d/2$, we obtain $c_1(\frac{7}{8}d^4 - 4d^2)\beta^2 + c_2(\frac{15}{16}d^5 - 2d^4)\beta$ from the left hand side, which is positive for dimension $d > 2$. This indicates that, in theory, when dealing with a knapsack-type basis, one can always achieve a better complexity by cutting the basis into two halves and pre-process them independently. This leads to the recursive reduction in the next section.

3.2 Recursive Reduction with LLL-Type Algorithms

The main idea is to apply our methodology to an input basis recursively, until one arrives to sub-lattice basis with dimension 2. In doing so, we achieve a upper bounded complexity of $O(d^{a_1-b_1}\beta^{b_1} + d^{a_2-b_2}\beta^{b_2})$. For simplicity, we deal with lattice whose dimension equals to a power of 2, although same principle is applicable to lattices with arbitrary dimensions.

Algorithm. We now describe our recursive reduction algorithm with LLL-type reduction algorithms. Let $LLL(\cdot)$ an LLL reduction algorithm that for any lattice basis B, it returns a reduced basis B_r. Algorithm 1 describes our algorithm, where B is a knapsack-type basis of a d-dimensional lattice, and d is a power of 2.

Since we have proven that, for any dimension of knapsack-type basis, it is always better to reduce its sub-lattice in advance as long as Equation 3 holds, it is straightforward to draw the following conclusion: the best complexity to reduce a knapsack-type basis with LLL-type reduction algorithms occurs when one cuts the basis recursively until one arrives with dimension 2 sub-lattices.

Algorithm 1. Recursive Reduction with LLL algorithm

Input: B, d
Output: B_r
  $number\_of\_rounds \leftarrow \log_2 d$
  $B_b \leftarrow B$
  for $i \leftarrow 1 \rightarrow number\_of\_rounds$ **do**
    $dim\_of\_sublattice \leftarrow 2^i$
    $number\_of\_blocks \leftarrow d/dim\_of\_sublattice$
    $B_r \leftarrow EmptyMatrix()$
    for $j \leftarrow 1 \rightarrow number\_of\_blocks$ **do**
      $B_t \leftarrow SubMatrix(B_b, (j-1)*dim\_of\_sublattice+1, 1, j*dim\_of\_sublattice, d)$
      $B_t \leftarrow LLL(B_t)$
      $B_r \leftarrow VerticalJoin(B_r, B_t)$
    end for
    $B_b \leftarrow B_r$
  end for

In Algorithm 1, $EmptyMatrix(\cdot)$ is to generate a 0 by 0 matrix; $SubMatrix(B, a, b, c, d)$ is to extract a matrix from B, starting from a-th row and b-th column, finishing at c-th row and d-th column; while $VerticalJoin(A, B)$ is to adjoin two matrix with same number of columns vertically.

Complexity. In the following, we prove that the complexity of our algorithm is $O(d^{a_1-b_1}\beta^{b_1} + d^{a_2-b_2}\beta^{b_2})$, assuming $\rho < 1$.

For the i-th round, to reduce a single block takes $c_1 2^{ia_1}(\frac{\beta}{2^{i-1}})^{b_1} + c_2 2^{ia_2}(\frac{\beta}{2^{i-1}})^{b_2}$, while there exist $\frac{d}{2^i}$ such blocks. Hence, the total complexity is as follows:

$$
\begin{aligned}
&\sum_{i=1}^{\log_2 d}\left(\frac{d}{2^i}\right)(c_1 2^{ia_1}(\beta/2^{i-1})^{b_1} + c_2 2^{ia_2}(\beta/2^{i-1})^{b_2}) \\
&\quad = d\cdot\sum_{i=1}^{\log_2 d}(c_1 2^{i(a_1-b_1-1)+b_1}\beta^{b_1} + c_2 2^{i(a_2-b_2-1)+b_2}\beta^{b_2}) \\
&\quad = c_1 2^{b_1} d\beta^{b_1}\left(\sum_{i=1}^{\log_2 d} 2^{(a_1-b_1-1)i}\right) + c_2 2^{b_2} d\beta^{b_2}\left(\sum_{i=1}^{\log_2 d} 2^{(a_2-b_2-1)i}\right) \\
&\quad < c_1 2^{b_1} d\beta^{b_1}\left(2^{(\log_2 d+1)(a_1-b_1-1)}\right) + c_2 2^{b_2} d\beta^{b_2}\left(2^{(\log_2 d+1)(a_2-b_2-1)}\right) \\
&\quad < c_1 2^{b_1} d\beta^{b_1}(2d)^{a_1-b_1-1} + c_2 2^{b_2} d\beta^{b_2}(2d)^{a_2-b_2-1} \\
&\quad < c_1 2^{a_1-1} d^{a_1-b_1}\beta^{b_1} + c_2 2^{a_2-1} d^{a_2-b_2}\beta^{b_2}.
\end{aligned}
$$

As a result, we obtain a new time complexity $O(d^{a_1-b_1}\beta^{b_1} + d^{a_2-b_2}\beta^{b_2})$.

4 Applying Recursive Reduction to L^2

We adapt the classic L^2 as an example. The L^2 algorithm uses a worst-case complexity of $c_1 d^4 \beta^2 + c_2 d^5 \beta$ for arbitrary basis. Therefore, applying our recursive methodology, one obtains

$$\begin{aligned}
&\sum_{i=1}^{\log_2 d} \left(\frac{d}{2^i}\right) \left(c_1 2^{4i} \left(\frac{\beta}{2^{i-1}}\right)^2 + c_2 2^{5i} \left(\frac{\beta}{2^{i-1}}\right)\right) \\
&\qquad = \sum_{i=1}^{\log_2 d} (4c_1 d 2^i \beta^2 + 2c_2 d 2^{3i} \beta) \\
&\qquad = 4c_1 d\beta^2 \left(\sum_{i=1}^{\log_2 d} 2^i\right) + 2c_2 d\beta \left(\sum_{i=1}^{\log_2 d} 2^{3i}\right) \\
&\qquad < 4c_1 d\beta^2 (2d) + 2c_2 d\beta 1.15 d^3 \\
&\qquad < 8c_1 d^2 \beta^2 + 2.3 c_2 d^4 \beta.
\end{aligned}$$

Now we compare our complexity with the original L^2 algorithm. As mentioned earlier, when applying to a knapsack-type basis, the provable worst-case complexity of L^2 becomes $c_1 d^3 \beta^2 + c_2 d^4 \beta$ rather than $c_1 d^4 \beta^2 + c_2 d^5 \beta$ as for a random basis. However, it is worth pointing out that in practice, one can achieve a much better result than a worst case, since the weight of most X_i is equally distributed into all the columns. Heuristically, one can expect $\Theta(c_1 d^2 \beta^2)$ when d, β go to infinity and $\beta \gg d$.

Input a knapsack-type basis, the L^2 algorithm (and almost all other LLL-type reduction algorithms) tries to reduce the first k rows, then the $k+1$ row, $k+2$ row, etc. For a given $k+1$ step, the current basis has the following shape:

$$B_{knap-L^2} = \begin{pmatrix}
x_{1,1} & x_{1,2} & \dots & x_{1,k+1} & 0 & 0 & \dots & 0 \\
x_{2,1} & x_{2,2} & \dots & x_{2,k+1} & 0 & 0 & \dots & 0 \\
\vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots \\
x_{k,1} & x_{k,2} & \dots & x_{k,k+1} & 0 & 0 & \dots & 0 \\
X_{k+1} & 0 & \dots & 0 & 1 & 0 & \dots & 0 \\
X_{k+2} & 0 & \dots & 0 & 0 & 1 & \dots & 0 \\
\vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots \\
X_d & 0 & \dots & 0 & 0 & 0 & \dots & 1
\end{pmatrix}$$

L^2 will reduce the first $k+1$ rows during this step. Despite that most of the entries are with small elements ($\|x_{i,j}\| \sim O(2^{\frac{\beta}{k}})$), the worse-case complexity of current step still depends on the last row of current step, i.e., $\langle X_{k+1}, 0, \dots, 0, 1, 0, \dots, 0 \rangle$.

For the recursive reduction, on the final step, the input basis is in the form of:

$$B_{rec-L^2} = \begin{pmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,\frac{d}{2}+1} & 0 & 0 & \dots & 0 \\ x_{2,1} & x_{2,2} & \dots & x_{2,\frac{d}{2}+1} & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \dots & \vdots \\ x_{\frac{d}{2},1} & x_{k,2} & \dots & x_{\frac{d}{2},\frac{d}{2}+1} & 0 & 0 & \dots & 0 \\ x_{\frac{d}{2}+1,1} & 0 & \dots & 0 & x_{\frac{d}{2}+1,\frac{d}{2}+2} & x_{\frac{d}{2}+1,\frac{d}{2}+3} & \dots & x_{\frac{d}{2}+1,d+1} \\ x_{\frac{d}{2}+2,1} & 0 & \dots & 0 & x_{\frac{d}{2}+2,\frac{d}{2}+2} & x_{\frac{d}{2}+2,\frac{d}{2}+3} & \dots & x_{\frac{d}{2}+2,d+1} \\ \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \dots & \vdots \\ x_{d,1} & 0 & \dots & 0 & x_{d,\frac{d}{2}+2} & x_{d,\frac{d}{2}+3} & \dots & x_{d,d+1} \end{pmatrix}$$

Note that the weight of X_i is equally distributed into $\frac{d}{2}+1$ columns. Hence, the bit length of maximum norm of basis is reduced from β to approximately $d\log_2 c_0 + 2\beta/d$. Therefore, we achieve a better time complexity. In fact, the provable new complexity is of the same level of the heuristic results observed in practice, when $\beta \gg d$.

5 Special Case: Principal Ideal Lattice Basis

In this section, we present a technique when dealing with a principal ideal lattice basis. Due to the special form of a principal ideal lattice, we are able to reduce the number of reductions in each round to 1, with a cost of $O(d)$ additional vectors for the next round. This technique does not effect the asymptotic complexity, however, in practice, it will accelerate the reduction.

$$B_I = \begin{pmatrix} \delta & 0\,0\dots 0 \\ -\alpha \bmod \delta & 1\,0\dots 0 \\ -\alpha^2 \bmod \delta & 0\,1\dots 0 \\ \vdots & \vdots\,\vdots\,\ddots\,\vdots \\ -\alpha^{d-1} \bmod \delta & 0\,0\dots 1 \end{pmatrix} \Longrightarrow B'_I = \begin{pmatrix} \delta & 0 & 0\dots & 0 & 0 \\ -\alpha & 1 & 0\dots & 0 & 0 \\ 0 & -\alpha & 1\dots & 0 & 0 \\ \vdots & \vdots & \vdots\,\ddots & \vdots & \vdots \\ 0 & 0 & 0\dots & -\alpha & 1 \end{pmatrix}.$$

Let $X_0 = \delta$, one obtains B_I in the above form. From B_I, one constructs a new basis B'_I. Then, one can obtain a generator matrix of $\mathcal{L}(B_I)$ by inserting some vectors in $\mathcal{L}$ to B'_I.

The following example shows how to construct G with $d = 5$. In this example, since vector $\langle 0, 0, \delta, 0, 0\rangle$ is a valid vector in $\mathcal{L}(B)$, B and G span a same lattice. Applying a lattice reduction algorithm over G will return a matrix with the top row that is a zero vector, while the rest form a reduced basis of $\mathcal{L}$.

$$G = \left(\begin{array}{ccccc} \delta & 0 & 0 & 0 & 0 \\ -\alpha & 1 & 0 & 0 & 0 \\ 0 & -\alpha & 1 & 0 & 0 \\ \hline 0 & 0 & \delta & 0 & 0 \\ 0 & 0 & -\alpha & 1 & 0 \\ 0 & 0 & 0 & -\alpha & 1 \end{array}\right) \Longrightarrow \left(\begin{array}{ccccc} x_{1,1} & x_{1,2} & x_{1,3} & 0 & 0 \\ x_{2,1} & x_{2,2} & x_{2,3} & 0 & 0 \\ x_{3,1} & x_{3,2} & x_{3,3} & 0 & 0 \\ \hline 0 & 0 & x_{1,1} & x_{1,2} & x_{1,3} \\ 0 & 0 & x_{2,1} & x_{2,2} & x_{2,3} \\ 0 & 0 & x_{3,1} & x_{3,2} & x_{3,3} \end{array}\right)$$

To reduce G, we adopt our recursive reduction methodology. We firstly reduce the top half of G. Since the second half is identical to the top half, except the position of the elements, we do not need to reduce the second half. Indeed, we use the result of the top half block and then shift all the elements. In this case, during our recursive reduction, for round i, instead of doing $d/2^i$ reductions, one need to perform only one reduction. Finally, one reduces the final matrix G, removes all the zero vectors and start a new round.

With our technique, the number of vectors grows, and this may increase the complexity of the next round. For the i-th round, the number of vectors grows by $d/2^{i-1} - 1$. It will be negligible when $d/2^i \ll d$. For instance, if we adopt this approach between the second last round and the last round, this approach will only increase the number of vectors by 1, while if one uses it prior to the first round, the number of rows will almost be doubled. In practice, one can choose to adopt this technique for each round only when it accelerates the reduction.

We note that the asymptotic complexity remains the same, since generally speaking, the number of vectors remains $O(d)$ as before, while the asymptotic complexity concerns only d and β.

6 Conclusion

In this paper, we presented a methodology for lattice reduction algorithms used for solving low density modular knapsack problems. The complexity of polynomial time lattice reduction algorithms relies on the dimension d and the bit length β of maximum norm of input basis. We prove that for a knapsack-type basis, it is always better to pre-process the basis by distributing the weight to many columns as equally as possible. Using this methodology recursively, we are able to reduce β to approximately $2\beta/d$, and consequently, we successfully reduce the entire complexity.

We then demonstrated our technique over the floating-point LLL algorithm. We obtain a provable upper bounded complexity of $O(d^{2+\varepsilon}\beta^2 + d^{4+\varepsilon}\beta)$, which is by far the best provable time complexity for a knapsack-type basis.

References

1. Ajtai, M.: The shortest vector problem in l_2 is NP-hard for randomized reductions (extended abstract). In: Thirtieth Annual ACM Symposium on the Theory of Computing (STOC 1998), pp. 10–19 (1998)
2. Chen, Y., Nguyên, P.Q.: BKZ 2.0: Better Lattice Security Estimates. In: Lee, D.H. (ed.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 1–20. Springer, Heidelberg (2011)
3. Coster, M.J., Joux, A., LaMacchia, B.A., Odlyzko, A.M., Schnorr, C.-P., Stern, J.: Improved low-density subset sum algorithms. Computational Complexity 2, 111–128 (1992)
4. Gama, N., Nguyên, P.Q.: Predicting Lattice Reduction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 31–51. Springer, Heidelberg (2008)
5. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) STOC, pp. 169–178. ACM (2009)

6. Goldstein, D., Mayer, A.: On the equidistribution of Hecke points. Forum Mathematicum 15, 165–189 (2006)
7. Hanrot, G., Pujol, X., Stehlé, D.: Analyzing Blockwise Lattice Algorithms Using Dynamical Systems. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 447–464. Springer, Heidelberg (2011)
8. Kannan, R.: Improved algorithms for integer programming and related lattice problems. In: Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing, STOC 1983, pp. 193–206. ACM, New York (1983)
9. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) Complexity of Computer Computations. The IBM Research Symposia Series, pp. 85–103. Plenum Press, New York (1972)
10. Koy, H., Schnorr, C.-P.: Segment LLL-Reduction of Lattice Bases. In: Silverman, J.H. (ed.) CaLC 2001. LNCS, vol. 2146, pp. 67–80. Springer, Heidelberg (2001)
11. Koy, H., Schnorr, C.-P.: Segment LLL-Reduction with Floating Point Orthogonalization. In: Silverman, J.H. (ed.) CaLC 2001. LNCS, vol. 2146, pp. 81–96. Springer, Heidelberg (2001)
12. Lagarias, J.C., Odlyzko, A.M.: Solving low-density subset sum problems. J. ACM 32(1), 229–246 (1985)
13. Lai, M.K.: Knapsack cryptosystems: The past and the future (2001)
14. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen 261, 513–534 (1982)
15. Lovász, L.: An Algorithmic Theory of Numbers, Graphs and Convexity. In: CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 50. SIAM Publications (1986)
16. Micciancio, D., Goldwasser, S.: Complexity of Lattice Problems, A Cryptographic Perspective. Kluwer Academic Publishers (2002)
17. Minkowski, H.: Geometrie der Zahlen. B. G. Teubner, Leipzig (1896)
18. Morel, I., Stehlé, D., Villard, G.: H-LLL: using householder inside LLL. In: ISSAC, pp. 271–278 (2009)
19. Nguyên, P.Q., Stehlé, D.: Floating-Point LLL Revisited. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 215–233. Springer, Heidelberg (2005)
20. Nguyên, P.Q., Stehlé, D.: LLL on the Average. In: Hess, F., Pauli, S., Pohst, M. (eds.) ANTS 2006. LNCS, vol. 4076, pp. 238–256. Springer, Heidelberg (2006)
21. Nguyên, P.Q., Stern, J.: Adapting Density Attacks to Low-Weight Knapsacks. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 41–58. Springer, Heidelberg (2005)
22. Novocin, A., Stehlé, D., Villard, G.: An LLL-reduction algorithm with quasi-linear time complexity: extended abstract. In: Fortnow, L., Vadhan, S.P. (eds.) STOC, pp. 403–412. ACM (2011)
23. Pujol, X., Stehlé, D., Cade, D.: fplll library, `http://perso.ens-lyon.fr/xavier.pujol/fplll/`
24. Schnorr, C.-P.: A more efficient algorithm for lattice basis reduction. J. Algorithms 9(1), 47–62 (1988)
25. Schnorr, C.-P.: Fast LLL-type lattice reduction. Inf. Comput. 204(1), 1–25 (2006)
26. Smart, N.P., Vercauteren, F.: Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes. In: Nguyên, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 420–443. Springer, Heidelberg (2010)
27. van Hoeij, M., Novocin, A.: Gradual sub-lattice reduction and a new complexity for factoring polynomials. Algorithmica 63(3), 616–633 (2012)