A Complexity Analysis of a Jacobi Method for Lattice Basis Reduction

Zhaofei Tian
Department of Computing and Software, McMaster University
Hamilton, Ontario, L8S 4K1, Canada
Sanzheng Qiao
Department of Computing and Software, McMaster University
Hamilton, Ontario, L8S 4K1, Canada

Abstract

The famous LLL algorithm is the first polynomial time lattice reduction algorithm which is widely used in many applications. In this paper, we prove the convergence of a novel polynomial time lattice reduction algorithm, called the Jacobi method introduced by S. Qiao [23], and show that it has the same complexity as the LLL algorithm. Our experimental results show that the Jacobi method outperforms the LLL algorithm in not only efficiency, but also orthogonality defect of the bases it produces.

Categories and Subject Descriptors

General Terms Algorithms and Theory

Keywords Lattice, basis, Jacobi method, Lagrange algorithm, lattice basis reduction, LLL.

1 Introduction

In recent years, lattice based methods are more and more involved in many applications, such as cryptography [9, 3], Global Positioning System (GPS) [27, 7] and wireless communications [26]. A *lattice* is a set of discrete points in Euclidean Space, represented by integer linear combinations of a set of linearly independent vectors [16]. The set of independent vectors that generate the lattice is called a *basis* for the lattice. The number of vectors in a basis is defined as the *dimension* of the lattice.

Given a lattice L of dimension n ($n \geq 2$), its basis is not unique. In fact, it has infinitely many bases [9]. The *Lattice Reduction Problem* is to find the bases whose vectors are more orthogonal to each other (or shorter) than the given basis, depending on different measurements.

Minkowski introduced a reduced basis [18, 4, 13] in 1890's, which is now called *Minkowski reduced basis*. Minkowski reduced basis is the strongest definition for lattice bases. It requires the shortest vectors that can form a basis for the lattice. A weaker notation called HKZ reduced basis is introduced in 1870's by Korkine and Zolotareff [14], which is based on Hermite's first formal definition of lattice reduction in 1850's.

There are many lattice reduction algorithms. Roughly speaking, the lattice reduction algorithms can be grouped into two catagories according to their complexity : exponential time algorithms and polynomial time algorithms. There are no known polynomial time algorithms for producing Minkowski reduced bases and HKZ reduced bases [1]. In 1983, R. Kannan gave an algorithm for constructing HKZ reduced bases [12]. The algorithm was further refined to construct both Minkowski reduced bases and HKZ reduced bases by B. Helfrich [8] in 1985 and R. Kannan in 1987 [13]. Practical algorithms for computing Minkowski reduced and HKZ reduced lattice bases can be found in [25], which use sphere decoding strategies to find a shortest vector in a lattice.

The first polynomial time lattice reduction algorithm was presented in 1982 [15], known as *LLL*

reduced bases , named after the three authors A. Lenstra, W. Lenstra, and L. Lovász. Theoretically, the LLL reduction algorithm can produce an approximate shortest vector that is at most a factor of $O(2^n)$ times longer than a real shortest vector of the lattice [2]. However, it performs well in practice. This extraordinary practical performance of LLL algorithm makes the algorithm an important tool in cryptography [20], sphere decoding [22], integer programming and other applications [21]. There are also many other algorithms based on LLL algorithm, improving the original LLL algorithm in either efficiency or accuracy.

The *Jacobi lattice reduction algorithm* (short as *Jacobi method* in the following sections) presented in [23] adopts a different strategy from the LLL algorithm to construct a reduced basis in polynomial time. It embeds the *Lagrange reduction algorithm* [19] to reduce every pair of two vectors and produces a reasonably good basis for the given lattice. Another advantage of Jacobi method is that it inherits parallel computing. Therefore, the algorithm can be implemented on multi-processor computers to achieve high performance. Out experimental results show that the Jacobi method without parallel computing outperforms the LLL algorithm in both orthogonality defect and running time.

The paper is organized as follows. In this section, we give several basic concepts about lattice, bases and lattice reduction algorithms. In the next section, the Lagrange reduction algorithm and Jacobi reduction algorithm are presented. The convergence proof and complexity analysis of Jacobi method are given in section 3. In section 4, we compare the Jacobi method with widely used LLL algorithm [15] in terms of orthogonality defect (or Hadamard Ratio) and running time. Finally, the paper is concluded in section 5.

2 Jacobi Method for Lattice Basis Reduction

In this section, we recall in detail the Lagrange reduction algorithm [19] and the Jacobi method for lattice basis reduction [23]. A unimodular transformation [25] is also recalled in this section, which will be used in the Lagrange algorithm and the Jacobi method.

2.1 Unimodular Transformation

Given a lattice L and a basis matrix A, we know there are more than one bases for L. Those bases have a property that the *volume*s of the parallelepipeds they generate are equal [16, 9]. We call the volume of those parallelepipeds the *determinant* of the lattice L.

Assume B is another basis matrix for the lattice L, then we have $|\det(A)| = |\det(B)|$. For the two basis matrices A and B for the given lattice, we can always find an integer matrix Z, called a *unimodular* matrix, such that $B = AZ$. Since B is a basis matrix for L, the columns in A can be represented by the integer linear combinations of columns in B. Hence, there exists another integer matrix $\bar{Z}$ such that $A = B\bar{Z}$. Obviously, $\bar{Z}$ is the inverse of Z. Since $|\det(A)| = |\det(B)|$, we know that $\det(Z) = \pm 1$.

Definition 2.1 (Unimodular)**.** A nonsigular integer matrix M is called *Unimodular* if $\det(M) = \pm 1$.

The purpose of lattice reduction algorithm is to find a unimodular matrix that transforms the given basis into a more orthogonal one.

Since there are more than one basis for a lattice of dimension greater than or equal to 2, it is necessary to define a measurement of the orthogonality of a basis. In this paper, we adopt the measurement called *Hadamard Ratio* [9] or *orthogonal defect*.

Definition 2.2 (Hadamard Ratio [9])**.** Given a basis matrix $A = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n]$ for a lattice L, the *Hadamard Ratio* of the basis matrix A is defined as

$$\mathcal{H}(A) = \left(\frac{\det(L)}{||\mathbf{a}_1||_2 \cdot ||\mathbf{a}_2||_2 \cdots ||\mathbf{a}_n||_2}\right)^{\frac{1}{n}} \quad . \tag{2.1}$$

According to the *Hadamard's Inequality* [10], we always have $\det(L) \leq ||\mathbf{a}_1||_2 \cdot ||\mathbf{a}_2||_2 \cdots ||\mathbf{a}_n||_2$, hence $0 < \mathcal{H}(A) \leq 1$, and the equality holds if and only if $\mathbf{a}_i$ $(1 \leq i \leq n)$ are orthogonal to each other. Geometrically, the Hadamard Ratio measures the scaled geometric mean of the lengths of columns in A. Therefore, the shorter the mean of the columns is, the closer the Hadamard Ratio (2.1) is to 1, and the more orthogonal the columns in A are.

2.2 Lagrange Reduction Algorithm

Given two integers, the *Euclidean Algorithm* [24] computes their greatest common divisor in polynomial time. The Lagrange's lattice reduction algorithm [23] uses a similar idea. It computes an optimally (Minkowski) reduced basis for two dimensional lattices in polynomial time [5, 19].

Definition 2.3 (Lagrange Reduced Basis [23])**.** Given a two-dimensional lattice L and its basis matrix $A = [\mathbf{a}_1, \mathbf{a}_2]$, A is called *Lagrange-Reduced* (or *L-Reduced*), if it satisfies

$$||\mathbf{a}_1||_2 \leq ||\mathbf{a}_2||_2 \quad \text{and} \quad |\mathbf{a}_1^T \mathbf{a}_2| \leq \frac{1}{2}||\mathbf{a}_1||_2^2. \tag{2.2}$$

The Lagrange reduced basis is a Minkowski reduced basis in two dimensional case [20]. The angle of two vectors in the produced bases is located in $[\pi/3, 2\pi/3]$ [23]. Because if we denote θ the angle between $\mathbf{a}_1$ and $\mathbf{a}_2$, we have $|\cos(\theta)| = |\mathbf{a}_1^T \mathbf{a}_2|/(||\mathbf{a}_1||_2||\mathbf{a}_2||_2) \leq |\mathbf{a}_1^T \mathbf{a}_2|/||\mathbf{a}_1||_2^2 \leq 1/2$.

Given a lattice L and its basis matrix $A = [\mathbf{a}_1, \mathbf{a}_2]$, the following Lagrange reduction algorithm produces a unimodular matrix Z_{12}, such that $[\mathbf{a}_1, \mathbf{a}_2]Z_{12}$ forms a Lagrange reduced basis matrix for the lattice L.

Algorithm 1: Lagrange reduction algorithm for two dimensional lattices

Input : A lattice basis matrix $[\mathbf{a}_1, \mathbf{a}_2]$

Output: A Lagrange reduced basis matrix $[\mathbf{a}_1, \mathbf{a}_2]Z_{12}$

1 **if** $||\mathbf{a}_1||_2 < ||\mathbf{a}_2||_2$ **then**

2 $\quad$ **SWAP**$(\mathbf{a}_1, \mathbf{a}_2)$;

3 Set $q = \lfloor \mathbf{a}_1^T \mathbf{a}_2 / ||\mathbf{a}_2||_2^2 \rceil$;

4 $\mathbf{Z}_{12} = \begin{bmatrix} 0 & 1 \\ 1 & -q \end{bmatrix}$;

5 $[\mathbf{a}_1, \mathbf{a}_2] \leftarrow [\mathbf{a}_1, \mathbf{a}_2]\mathbf{Z}_{12}$;

6 **while** $||\mathbf{a}_1||_2 > ||\mathbf{a}_2||_2$ **do**

7 $\quad$ Set $q = \lfloor \mathbf{a}_1^T \mathbf{a}_2 / ||\mathbf{a}_2||_2^2 \rceil$;

8 $\quad$ $\mathbf{Z}_{12} = \begin{bmatrix} 0 & 1 \\ 1 & -q \end{bmatrix}$;

9 $\quad$ $[\mathbf{a}_1, \mathbf{a}_2] \leftarrow [\mathbf{a}_1, \mathbf{a}_2]\mathbf{Z}_{12}$;

When the algorithm terminates, the given basis matrix $[\mathbf{a}_1, \mathbf{a}_2]$ above is overwritten by a Lagrange reduced basis. The notation $\lfloor \cdot \rceil$ in line 3 represents the nearest integer rounding. The algorithm keeps reducing the longer vector $\mathbf{a}_1$, until it cannot be reduced.

The Lagrange algorithm is proved to be a greedy algorithm [19]. In each iteration, the locally optimal choice of the integer scalar q helps us to find a global optimum of the two vectors reduced. Thus the algorithm produces a Minkowski reduced basis.

2.3 Jacobi Reduction Algorithm

The Jacobi method is originally proposed by C. Jacobi in 1846 for solving eigenvalue problems of real symmetric matrices [11, 6]. The method iteratively performs a singular value decomposition on every 2×2 submatrix until the symmetric matrix becomes almost diagonal. A Jacobi method

for lattice basis reduction introduced by S. Qiao [23] embeds the Lagrange algorithm as a tool worked on every 2×2 submatrix to reduce every pair of vectors. Hence it produces a reduced basis for the given lattice.

For a given general n dimensional lattice, the Jacobi method produces a reduced basis defined as follows.

Definition 2.4 ([23])**.** Given an n dimensional lattice L and its basis matrix $A = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n]$, A is called reduced if

$$||\mathbf{a}_i||_2 \leq ||\mathbf{a}_j||_2 \quad \text{and} \quad |\mathbf{a}_i^T \mathbf{a}_j| \leq \frac{1}{2}||\mathbf{a}_i||_2^2, \quad \text{for all } 1 \leq i < j \leq n. \tag{2.3}$$

It is not hard to see that the vectors in the above reduced basis are sorted by their lengths. The definition focuses on every pair of vectors locally. We introduce a modified Lagrange reduction algorithm [23] first as a tool to reduce each pair of vectors. It works on every 2×2 submatrix of a given lattice basis matrix $A = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n]$. *Gram Matrix* $G = A^T A$ is used in the following algorithm. The elements in the Gram matrix $G = [g_{ij}]$ have some special properties, such as $g_{ij} = g_{ji} = \mathbf{a}_i^T \mathbf{a}_j$ and $g_{ii} = ||\mathbf{a}_i||_2^2$.

The following algorithm runs the Lagrange reduction algorithm on a given pair (i, j) of indices.

Algorithm 2: Lagrange2(G, i, j)

Input : A *Gram* matrxi $G = A^T A$ and two indices i, j

Output: An unimodular matrix Z_{ij} such that the pair of ith, jth vectors in AZ_{ij} is L-reduced

$Z_{ij} = I_n$;

if $g_{ii} < g_{jj}$ **then**

  Swap $G(:,i)$ and $G(:,j)$;

  Swap $G(i,:)$ and $G(j,:)$;

  Swap $Z(:,i)$ and $Z(:,j)$;

Set $q = \lfloor g_{ij}/g_{jj} \rceil$;

Set Z to be I_n except $z_{ii} = 0, z_{jj} = -q$ and $z_{ij} = z_{ji} = 1$;

$G \leftarrow Z^T GZ$;

$Z_{ij} \leftarrow Z_{ij} Z$;

while $g_{ii} > g_{jj}$ **do**

  Set $q = \lfloor g_{ij}/g_{jj} \rceil$;

  Set Z to be I_n except $z_{ii} = 0, z_{jj} = -q$ and $z_{ij} = z_{ji} = 1$;

  $G \leftarrow Z^T GZ$;

  $Z_{ij} \leftarrow Z_{ij} Z$;

Using Algorithm-2 as a reduction tool in 2×2 submatrices, the following algorithm (3) introduced by S. Qiao [23] computes a reduced basis defined in Definition-2.4.

Algorithm 3: Jacobi reduction algorithm

Input : A basis matrix $A = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n]$

Output: An unimodular matrix $\mathbf{Z}$ such that $A\mathbf{Z}$ forms a Jacobi reduced basis

1 $\mathbf{Z} = I_n$;

2 $G = A^T A$;

3 **while** *not all pairs* $(\mathbf{a}_i, \mathbf{a}_j)$ *satisfy the Jacobi reduced condition*-(2.3) **do**

4 **for** $i \leftarrow 1$ **to** $n-1$ **do**

5 **for** $j \leftarrow i+1$ **to** n **do**

6 $[\mathbf{G}, \mathbf{Z}_{ij}] \leftarrow$ Lagrange2($\mathbf{G}, i, j$) ;

7 $\mathbf{Z} \leftarrow \mathbf{Z}\mathbf{Z}_{ij}$;

Given a lattice basis matrix A, the Algorithm-3 computes a Gram matrix G and applies Lagrange reduction algorithm to all possible pairs of the basis in the **`while`** loop from line 3 to 7. A proof of convergence and a complexity analysis of the algorithm is presented in the next section.

3 Convergence and Complexity Analysis of Jacobi Method

In the first part of this section, we present a formal proof of the convergence of the Lagrange reduction algorithm. In the second part, we analyze the complexity of the Jacobi method based on the proof of the convergence in the first part, and show that the complexity of the Jacobi method is the same as the LLL algorithm. Our experimental results in the next section show that the Jacobi method is significantly faster than the LLL algorithm in practice.

3.1 Convergence of the Jacobi method

Since the Jacobi method is based on Lagrange reduction algorithm, we first prove the convergence of the Lagrange algorithm. The proof is partially based on some ideas in [5][Lemma 17.1.9] and

[19] and the behaviour analysis of the Euclidean Algorithm [24].

Given two vectors $\mathbf{a}_1$ and $\mathbf{a}_2$ of size n, we assume $||\mathbf{a}_1||_2 \geq ||\mathbf{a}_2||_2$ without loss of generality. Let $(\mathbf{a}_1', \mathbf{a}_2')$ and $(\mathbf{a}_1'', \mathbf{a}_2'')$ be the vectors produced after one and two iterations of the Lagrange algorithm, respectively. The key point in the proof is that the situation $||\mathbf{a}_2'||_2 > \frac{1}{\sqrt{3}}||\mathbf{a}_1||_2$ will occur only in the first or the last two iterations in the Lagrange algorithm. All other iterations lead to the result $||\mathbf{a}_2'||_2 \leq \frac{1}{\sqrt{3}}||\mathbf{a}_1||_2$. Hence we can safety use the factor $\sqrt{3}$ in proving convergence and evaluating complexity.

Proposition 3.1. *The Lagrange reduction algorithm-*(2) **`Lagrange2(`**G, i, j**`)`** *is convergent, and each iteration step will reduce the vectors with a factor of at least* $\sqrt{3}$ *except the first and the last two iterations.*

Proof. Given two vectors $\mathbf{a}_1$ and $\mathbf{a}_2$ (assume $||\mathbf{a}_1||_2 \geq ||\mathbf{a}_2||_2$), let $\mu = \frac{\mathbf{a}_1^T \mathbf{a}_2}{||\mathbf{a}_2||_2^2}$, we have the integer scalar

$$q = \lfloor \mu \rceil = \mu - \varepsilon, \quad \text{where } |\varepsilon| \leq 1/2. \tag{3.1}$$

Assume the Lagrange algorithm does not terminate after the first iteration, the vectors produced after an iterations are:

$$\texttt{t1}: \begin{cases} \mathbf{a}_1' = \mathbf{a}_2 \\ \mathbf{a}_2' = \mathbf{a}_1 - q \cdot \mathbf{a}_2 \end{cases}, \quad \text{where } ||\mathbf{a}_2'||_2 \leq ||\mathbf{a}_2||_2.$$

For convenience, we call this iteration `t1` in the rest of the proof. Then we have the following cases according to the possible values of the integer scalar q. The values of q will lead the algorithm to different branches in the next iteration.

- Case 1: $q = 0$

  There is no further reduction needed. Hence the Lagrange reduction algorithm terminate after `t1`.

- Case 2: $|q| \geq 2$

We show the iteration reduces the vectors by a factor of at least $\sqrt{3}$ in this case. Let

$$\mathbf{a}_2^* = \mathbf{a}_1 - \mu \mathbf{a}_2, \tag{3.2}$$

we know that $\mathbf{a}_2^*$ is orthogonal to $\mathbf{a}_2$ according to Equation-(3.1) and the Gram–Schmidt method [10], as the following Figure-(1) shows.

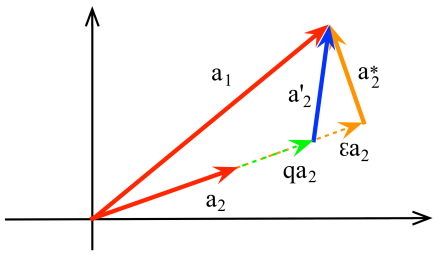

Figure 1: Single step Lagrange reduction

Thus we have $\mathbf{a}_2^{'} = \mathbf{a}_2^* + \varepsilon \mathbf{a}_2$, which means

$$||\mathbf{a}_2^{'}||_2^2 \leq ||\mathbf{a}_2^*||_2^2 + \frac{1}{4}||\mathbf{a}_2||_2^2 \,, \tag{3.3}$$

since $|\varepsilon| \leq 1/2$.

Therefore we have

$$\begin{aligned}
&\mathbf{a}_1 = \mu \mathbf{a}_2 + \mathbf{a}_2^* \\
\Rightarrow\ &\mathbf{a}_1 = (q+\varepsilon)\mathbf{a}_2 + \mathbf{a}_2^* \\
\Rightarrow\ &||\mathbf{a}_1||_2^2 = (q+\varepsilon)^2||\mathbf{a}_2||_2^2 + ||\mathbf{a}_2^*||_2^2 \quad \text{(since } \mathbf{a}_2 \text{ is orthogonal to } \mathbf{a}_2^*\text{)} \\
\Rightarrow\ &||\mathbf{a}_1||_2^2 \geq \frac{9}{4}||\mathbf{a}_2||_2^2 + ||\mathbf{a}_2^*||_2^2 \\
\Rightarrow\ &||\mathbf{a}_1||_2^2 \geq 2||\mathbf{a}_2||_2^2 + \frac{1}{4}||\mathbf{a}_2||_2^2 + ||\mathbf{a}_2^*||_2^2 \\
\Rightarrow\ &||\mathbf{a}_1||_2^2 \geq 2||\mathbf{a}_2||_2^2 + ||\mathbf{a}_2^{'}||_2^2 \quad \text{(since (3.3))} \\
\Rightarrow\ &||\mathbf{a}_1||_2^2 \geq 3||\mathbf{a}_2^{'}||_2^2 \quad \text{(since } \ ||\mathbf{a}_2||_2 \geq ||\mathbf{a}_2^{'}||_2\text{)}
\end{aligned}$$

Thus, the factor $\sqrt{3}$ holds for this case.

- Case 3: $q = 1$

  In this case, we show the Lagrange algorithm either reduces the vectors with a factor of at least $\sqrt{3}$, or terminates in the following iteration.

  If $||\mathbf{a}_2'||_2 \leq \frac{1}{\sqrt{3}}||\mathbf{a}_1||_2$, we have gotten the reduced factor $\sqrt{3}$ for this case. Thus it suffices to show that if

  $$||\mathbf{a}_2'||_2 > \frac{1}{\sqrt{3}}||\mathbf{a}_1||_2 \geq \frac{1}{\sqrt{3}}||\mathbf{a}_2||_2 \,, \tag{3.4}$$

  the algorithm terminates in the following iteration. Now we assume the Lagrange algorithm doesn't terminate in `t1`. Then the vectors produced in `t1` and the next iteration, we call it `t2`, are

  $$\texttt{t1}: \begin{cases} \mathbf{a}_1' = \mathbf{a}_2 \\ \mathbf{a}_2' = \mathbf{a}_1 - \mathbf{a}_2 \end{cases} \quad \text{and} \quad \texttt{t2}: \begin{cases} \mathbf{a}_1'' = \mathbf{a}_2' \\ \mathbf{a}_2'' = \mathbf{a}_1' - q' \cdot \mathbf{a}_2' \end{cases},$$

  where

  $$||\mathbf{a}_2'||_2 \leq ||\mathbf{a}_2||_2 \quad \text{and} \quad ||\mathbf{a}_2''||_2 \leq ||\mathbf{a}_2'||_2. \tag{3.5}$$

  To determine the value of the integer scalar q' in `t2`, we have

  $$\begin{aligned} |\mu| &= \frac{|\mathbf{a}_1'^T \mathbf{a}_2'|}{||\mathbf{a}_2'||_2^2} \\ &= \frac{|\mathbf{a}_2^T(\mathbf{a}_2^* + \varepsilon \mathbf{a}_2)|}{||\mathbf{a}_2'||_2^2} \quad (\mathbf{a}_2^* \text{ is defined in (3.2)}) \\ &= \frac{\varepsilon ||\mathbf{a}_2||_2^2}{||\mathbf{a}_2'||_2^2} \quad (\text{since } \mathbf{a}_2 \text{ is orthogonal to } \mathbf{a}_2^*) \\ &< \frac{3}{2} \quad (\text{since the assumption(3.4) and } |\varepsilon| \leq 1/2). \end{aligned}$$

  Therefore, the only possible values for q' are $-1, 0$ and 1, which gives us the following cases for `t2`. We prove that all the following cases will lead the algorithm to terminate. Therefore, if the reduced factor $\sqrt{3}$ does not hold, the Lagrange algorithm terminates in `t2`.

  - If $q' = 0$

    This means the vector $\mathbf{a}_2$ cannot be reduced. Hence the Lagrange algorithm terminates in `t2`.

– If $q' = -1$

  Substituting $\mathbf{a}'_1$ and $\mathbf{a}'_2$ in the iteration `t1` into the vectors produced in iteration `t2`, $\mathbf{a}''_2$ can be simplified as

$$\begin{aligned}\mathbf{a}''_2 &= \mathbf{a}'_1 - q' \cdot \mathbf{a}'_2 \\ &= \mathbf{a}_2 - q' \cdot (\mathbf{a}_1 - \mathbf{a}_2) \\ &= -q' \cdot \mathbf{a}_1 + (1 + q') \cdot \mathbf{a}_2 \\ &= \mathbf{a}_1 .\end{aligned} \tag{3.6}$$

  This is obviously a contradiction. Hence the Lagrange algorithm should terminates before `t2`.

– If $q' = 1$

  Substituting q' into Equation-(3.6), we get $\mathbf{a}''_2 = -\mathbf{a}_1 + 2\mathbf{a}_2$. According to inequation-(3.5), we have

$$\begin{aligned}&||\mathbf{a}''_2||_2 \leq ||\mathbf{a}'_2||_2 \\ \Rightarrow &||-\mathbf{a}_1 + 2\mathbf{a}_2||_2 \leq ||\mathbf{a}_1 - \mathbf{a}_2||_2 \\ \Rightarrow &||\mathbf{a}_1 - 2\mathbf{a}_2||_2 \leq ||\mathbf{a}_1 - \mathbf{a}_2||_2\end{aligned} \tag{3.7}$$

  If the equality in (3.7) holds, that is $||\mathbf{a}''_2||_2 = ||\mathbf{a}'_2||_2 = ||\mathbf{a}''_1||_2$, the Lagrange terminates, since the **`while`** loop condition in Algorithm-2 is $false$.

  We prove, by contradiction, that the inequality in (3.7) does not hold. If the inequality holds, the value of q in `t1` should have been 2, since the Lagrange algorithm is a greedy algorithm (as the section 2.2 shows), which always chooses the locally optimal q in each iteration. In this case, q equals 2 will produce a shorter $\mathbf{a}'_2$ than q equals 1. Hence it is a contradiction against our assumption.

Therefore, case $q = 1$ tells us that the Lagrange algorithm either reduces the vectors with a factor of at least $\sqrt{3}$, or terminates in the following iteration.

- Case 3: $q = -1$

  Similar with the previous case $q = 1$, we can arrive at the conclusion that the Lagrange algorithm either reduces the vectors with a factor of at least $\sqrt{3}$, or terminates in the following iteration.

To sum the proof up, each iterations of the Lagrange reduction algorithm reduces the two vectors with a factor of at least $\sqrt{3}$, except the first and last two iterations before the algorithm terminates.

□

3.2 Complexity analysis of the Jacobi method

To give precise running time estimations of the Jacobi Method, we first analyze the complexity of the Jacobi method globally with two main issues and several trivial parts. The two main issues are the complexity of a single iteration step in the Lagrange reduction algorithm and the maximum **`while`** loop rounds of the Jacobi Method.

1. Complexity of one single Lagrange iteration step

   The single step iteration of Lagrange reduction algorithm involves line 11 to 14 in the procedure **`Lagrange2(`**G, i, j**`)`**. It takes $O(1)$ to calculate and round the scalar q in line 6 and 11. Line 12 to 14 cost $O(n)$ arithmetic operations, since the algorithm implemented can use vector operations to update the matrix G and Z.

   Therefore, the total cost of one single Lagrange iteration is $O(n)$.

2. **`while`** loop analysis of the Jacobi Method

   The **`while`** loop starts at line 3 and ends at line 7 in algorithm **`Jacobi`**. Each round of **`while`** loop invokes the algorithm **`Lagrange2(`**G, i, j**`)`** at most $O(n^2)$ times. It has been proved in the previous section that every single iteration of the Lagrange reduction algorithm reduces a pair of vectors with a factor of at least $\sqrt{3}$. Since we use the Gram matrix $G = A^T A$ in the Jacobi Method, we can safely use the factor 3 in the complexity analysis of the **`while`**

loop.

Denote $B = \max_{1 \le i \le n}(g_{ii})$ and $D = \Pi_{i=1}^{n}(g_{ii})$, it is easy to check that $\lambda_1^{2n} \le D \le B^{2n}$, where λ_1 is the first *Minkowski's minima* [17] or the length of a shortest nonzero lattice vector. Then D is decreased at least a factor of 3 in each iteration of the Lagrange algorithm. Hence, we can calculate the maximum rounds of the **`while`** loop, which is $O(nlog_3\frac{B}{\lambda_1})$, or simply $O(nlogB)$. The interesting case is the lower bound of **`while`** loop rounds. It is $O(logB)$ in ideal situations, which indicates that the **`while`** loop will end in constant times ideally.

Therefore, the **`while`** loop involves a complexity factor of $O(n^3logB)$ for the worst case, and $O(n^2logB)$ under the ideal situation.

3. Computing the Gram Matrix G
   The computation of G takes $O(n^3)$ arithmetic operations.

To sum up, the Jacobi method takes an upper bound of $O(n^4logB)$ arithmetic operations, and ideal lower bound of $O(n^3logB)$ arithmetic operations. We can see the complexity of the Jacobi method is the same as the complexity of the LLL algorithm, which is also $O(n^4logB)$ [15]. According to our experimental data, the Jacobi method takes an average of 8 rounds for matrices of dimension less than 200. Hence, the Jacobi method most likely runs in lower bound for random matrices.

4 Experimental Result

Because of the extraordinary efficiency in practice, the LLL algorithm is widely used in many applications. To evaluate the practical efficiency of the Jacobi method, we compared the Jacobi method with the LLL algorithm. The two algorithms are implemented in **`MATLAB R2010b`** running on a **`Linux`** 32-bit version machine. The LLL algorithm is chosen as the vector-operated version [22] to avoid unnecessary operations and hence achieves high efficiency.

The dimension of experimental lattices varies as 10, 50, 100, 150 and 200. MATLAB function **`rand()`** is invoked to generate random lattice basis matrices for each dimension. In each chosen dimension, we run the two algorithms 100 times to get average result from the output.

	Hadamard Ratio		Runtime	
Dimention	Jacobi	LLL	Jacobi	LLL
10	0.7621	0.8458	0.0013	0.0064
50	0.4992	0.4650	0.0655	0.3565
100	0.4756	0.4221	0.2180	0.6512
150	0.4673	0.4181	0.3762	1.1870
200	0.4615	0.4158	0.4856	1.8557

Table 1: Hadamard Ratio and Runtime of Jacobi method and LLL algorithm

1. Orthogonality defact

   The experimental data shows the Jacobi method produces more orthogonal bases than the LLL algorithm for the lattices whose dimension is large than or equals 50.

2. Running time

   The data of running time also indicates the higher performance of the Jacobi method comparing with the LLL algorithm, which usually performs well in practical applications.

Our experimental data illustrates the advantages of the Jacobi method in both orthogonality defect and running time comparing with the widely used LLL algorithm. The Figure-(1) tells us that the Jacobi method is signaficantly faster than the LLL algorithm, which makes the Jacobi method a valuable algorithm for lattice basis reduction, especially the LLL algorithm is commonly regarded as the fastest lattice reduction algorithm.

5 Conclusion and Future Work

In this paper, we proved the convergence of a novel lattice reduction algorithm, called the Jacobi method [23], and evaluated its complexity. The complexity analysis shows that the Jacobi method has same complexity as the widely used LLL algorithm.

Our experimental results supported our analysis, which showed that the bases constructed by the Jacobi method were better than the bases produced by the LLL algorithm in terms of Hadamard Ratio, when the dimensions of lattices are greater than or equal 50. Besides, the Jacobi method is inherently parallel. The performance compare between Jacobi method and LLL algorithm illustrated that the former one is much faster than the other, even without parallel computing enabled. Therefore, the Jacobi method makes itself a better algorithm for lattice reduction required applications.

Further work includes more detailed investigation on performance, since our experimental data shows that the Jacobi method runs on the low bound of complexity for randomly generated matrices. Hence potential speedup of the Jacobi method is possible.

References

[1] Dorit Aharonov and Oded Regev. Lattice problems in NP and co-NP. *J. ACM*, 52:749–765, September 2005.

[2] L. Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6:1–13, 1986.

[3] Dan Boneh, The Rsa Cryptosystem, Invented Ron Rivest, Adi Shamir, Len Adleman, and Was Rst. Twenty years of attacks on the RSA cryptosystem. *Notices of the AMS*, 46:203–213, 1999.

[4] John L. Donaldson. Minkowski reduction of integral matrices. *j-MATH-COMPUT*, 33(145):201–216, jan 1979.

[5] Steven Galbraith. Mathematics of Public Key Cryptography. An unpublished edited version, April 2012.

[6] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 3rd edition, 1996.

[7] Babak Hassibi and Haris Vikalo. On the sphere-decoding algorithm i. expected complexity. *IEEE Trans. Sig. Proc*, pages 2806–2818, 2005.

[8] Bettina Helfrich. Algorithms to construct Minkowski reduced and Hermite reduced lattice bases. *Theoretical Computer Science*, 41:125 – 139, 1985.

[9] J. Hoffstein, J.C. Pipher, and J.H. Silverman. *An introduction to mathematical cryptography*. Undergraduate texts in mathematics. Springer, 2008.

[10] Roger A. Horn and Charles R. Johnson. *Matrix analysis*. Cambridge University Press, New York, NY, USA, 1986.

[11] C.J.G. Jacobi. Über ein leichtes verfahren, die in der theorie der säkularstörungen vorkommenden gleichungen numerisch aufzulösen. *Journal für reine und angewandte Mathematik*, 30:51–95, 1846.

[12] Ravi Kannan. Improved algorithms for integer programming and related lattice problems. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, STOC '83, pages 193–206, New York, NY, USA, 1983. ACM.

[13] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12:415–440, August 1987.

[14] A. Korkine and G. Zolotareff. Sur les formes quadratiques. *Mathematische Annalen*, 6:366–389, 1873. 10.1007/BF01442795.

[15] A.K. Lenstra, H.W.jun. Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982.

[16] Franklin T. Luk, Sanzheng Qiao, and Wen Zhang. A lattice basis reduction algorithm. Technical report, Institute for Computational Mathematics Hong Kong Baptist University, 2010.

[17] Romanos Malikiosis. An optimization problem related to Minkowski's successive minima. *Discrete Comput. Geom.*, 43:784–797, June 2010.

[18] H. Minkowski. Discontinuity region for arithmetical equivalence. *J. reine Angew*, (129):220–274, 1905.

[19] P. Q. Nguyen and D. Stehlé. Low-dimensional lattice basis reduction revisited. *ACM Transactions on Algorithms*, 2009. To appear.

[20] Phong Nguyen. Lattice reduction algorithms: Theory and practice. In Kenneth Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 2–6. Springer Berlin / Heidelberg, 2011.

[21] Phong Q. Nguyen and Brigitte Valle. *The LLL Algorithm: Survey and Applications*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[22] Sanzheng Qiao. Integer least squares: sphere decoding and the LLL algorithm. In *Proceedings of the 2008 C3S2E conference*, C3S2E '08, pages 23–28, New York, NY, USA, 2008. ACM.

[23] Sanzheng Qiao. A Jacobi method for lattice basis reduction. In *Proceedings of 2012 International Conference on Wireless Communications and Networks*, Xi'an China, May 2012. To appear.

[24] I. M. Vinogradov. *Elements of number theory*. Dover Publications Inc., New York, 1954. Translated by S. Kravetz.

[25] Sanzheng Qiao Wen Zhang and Yimin Wei. Practical algorithms for constructing HKZ and Minkowski reduced bases. Technical report, McMaster University, 2011. CAS-11-04-SQ.

[26] D. Wübben, D. Seethaler, J. Jaldén, and G. Matz. Lattice reduction: A survey with applications in wireless communications. *IEEE Signal Processing Magazine*, 28(3):70–91, May 2011.

[27] P. Xu, C. Shi, and J. Liu. Integer estimation methods for GPS ambiguity resolution: an applications oriented review and improvement. *Survey Review*, 44:59–71, Jan. 2012.