Article

N-Dimensional LLL Reduction Algorithm with Pivoted Reflection

Zhongliang Deng, Di Zhu * and Lu Yin

School of Electronic Engineering, Beijing University of Posts and Telecommunications, No. 10 Xitucheng Road, Beijing 100876, China; dengzhl@bupt.edu.cn (Z.D.); inlu_mail@163.com (L.Y.)

* Correspondence: cordialz@bupt.edu.cn; Tel.: +86-010-6119-8569

Received: 18 December 2017; Accepted: 14 January 2018; Published: 19 January 2018

Abstract: The Lenstra-Lenstra-Lovász (LLL) lattice reduction algorithm and many of its variants have been widely used by cryptography, multiple-input-multiple-output (MIMO) communication systems and carrier phase positioning in global navigation satellite system (GNSS) to solve the integer least squares (ILS) problem. In this paper, we propose an n-dimensional LLL reduction algorithm (n-LLL), expanding the Lovász condition in LLL algorithm to n-dimensional space in order to obtain a further reduced basis. We also introduce pivoted Householder reflection into the algorithm to optimize the reduction time. For an m-order positive definite matrix, analysis shows that the n-LLL reduction algorithm will converge within finite steps and always produce better results than the original LLL reduction algorithm with $n > 2$. The simulations clearly prove that n-LLL is better than the original LLL in reducing the condition number of an ill-conditioned input matrix with 39% improvement on average for typical cases, which can significantly reduce the searching space for solving ILS problem. The simulation results also show that the pivoted reflection has significantly declined the number of swaps in the algorithm by 57%, making n-LLL a more practical reduction algorithm.

Keywords: LLL reduction; pivoted reflection; integer least squares (ILS); global navigation satellite system (GNSS)

1. Introduction

With the rapid development of the Beidou System (BDS), the Galileo system, the Global Positioning System (GPS) and the GLONASS system, the Global Navigation Satellite System (GNSS) is serving more and more people with higher positioning accuracy [1,2]. Alongside the standard point positioning, carrier-phase based precise positioning techniques with cm-level to mm-level accuracy, such like real-time kinematic (RTK) and precise point positioning (PPP), have started to show their potential in public services other than in specific areas such as ground surveying. The commercial continuous operational reference station (CORS) network providers have enabled the precise positioning applications in unmanned aerial vehicle (UAV), unmanned autonomous vehicle and so on [3–5].

The main computational effort in carrier-phase precise positioning is to resolve the carrier phase integer ambiguity which is contaminated by all kind of noises during the signal propagation. Several efficient ambiguity resolution methods were proposed during the last several decades such as: Least-Square Ambiguity Searching Technique (LSAST) [6], Triple Frequency Ambiguity Resolution (TCAR) [7], Least-squares AMBiguity Decorrelation Adjustment (LAMBDA) [8,9]. The great breakthrough of LAMBDA algorithm is bringing the "decorrelation" process into integer ambiguity resolution, dividing the whole process into: estimation, decorrelation (also known as Z-transformation), search and back transformation.

As the measurements of pseudo-range and carrier phase are strongly correlated in time and space, the coefficient matrix to resolve the ambiguities is so ill-conditioned that the search space is huge and abnormal, making the search process extremely time-consuming and inefficient. For real-time applications, the decorrelation process is curial to reduce search effort. As for LAMBDA, integer Gauss transformation with permutation is used as the decorrelation process and is proven to be very effective. A series of algorithms have been applied to the decorrelation process since then. Xu used Cholesky decomposition to calculate the Z-transformation matrix [10]; Chang modified the Gauss transformation in LAMBDA with symmetric pivoting strategy [11]; Xu proposed the parallel Cholesky-based reduction method using minimum pivoting strategy [12] and Hassibi was the first to introduce LLL algorithm into integer ambiguity resolution [13].

The LLL reduction algorithm was first proposed by Arjen Lenstra, Hendrik Lenstra and László Lovász in 1982 [14] and had been proven useful polynomial-time algorithm to solve the closest vector problem (CVP) since then. With the development of lattice theory and its application, LLL algorithm becomes a powerful tool to solve the ILS problem, which expands its usage to numerous applications such like next generation MIMO communication detection algorithm [15–17], integer ambiguity resolution [13,18] and many other integer solution finding problem.

The original LLL reduction algorithm uses Gram-Schmidt orthogonalization to generate orthogonal basis, which involves O($n$log(B))-bit integer. A float point LLL (FPLLL) algorithm is proposed to avoid the waste of resources [19]. Schnorr used the half-k method [20] to ensure the calculation accuracy with FPLLL, which only involves O(n + log(B))-bit integer and converges in O(n^4log(B)). Koy proposed a segment LLL algorithm, weakening the constraints of reduction to ensure the efficiency of the algorithm when dealing with matrix of rank 350 or above. Schnoor proposed the deep insertion LLL (DeepLLL) and the Block Korkine-Zolotareff (BKZ) algorithm [21], which improved the performance significantly. Fontein made the DeepLLL algorithm a polynomial-time algorithm with the help of potential factor, naming it potential LLL (PotLLL) [22].

In this paper, we propose the n-LLL reduction, expanding the Lovász condition of original LLL algorithm to n-dimensional space. And we give out an adjustable parameter "n" to balance the performance and computational efforts. Pivoted Householder reflection and Givens transformation are also introduced into the n-LLL algorithm to further optimize the reduction time.

The performance and complexity of the n-LLL algorithm are then analyzed, showing that the new algorithm performs better than the original LLL algorithm with $n > 2$ and will always converge in polynomial time. The simulation results show that the n-LLL algorithm has better reduction quality than the original LLL with about 39% improvement on average. The new algorithm causes no significant increase in computational efforts because of the pivoted reflection, which is able to reduce as much as 57% swaps in the algorithm.

2. The LLL Reduction

A lattice is defined as:

$$L = \left\{ \sum_{i=1}^{m} a_i \mathbf{b}_i | a_i \in \mathbb{Z} \right\} \tag{1}$$

where $\mathbf{b}_i$ denote m independent liner vectors defined on $\mathbb{R}^n$ and are called a basis of lattice L. So, a lattice can be seen as a discrete point set inside the real value space $\mathbb{R}^n$.

Then the typical weighted ILS problem like:

$$\min_{\mathbf{a} \in \mathbb{Z}^n} \|\hat{\mathbf{a}} - \mathbf{a}\|^2_{Q_{\hat{a}}^{-1}} \tag{2}$$

can be described as: finding a vector $\mathbf{a} \in L$ that is closest to $\hat{\mathbf{a}} \in \mathbb{R}^n$ ($\mathbf{Q}_{\hat{a}}$ is the covariance matrix of vector $\hat{\mathbf{a}}$), which is also known as a CVP. As a CVP is an NP-hard problem, one efficient approach to acquire an approximate solution is lattice basis reduction. To simplify the demonstration, the discussions below involve only square matrix.

For a basis $\mathbf{B} = (\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3 \cdots \mathbf{b}_m)$ of lattice L, obviously $\mathbf{B}_o = \mathbf{B} \cdot \mathbf{T}$ is also a basis of L, where $\mathbf{T}$ is a unimodular matrix. Then the weighted ILS problem (2) can be rewritten as:

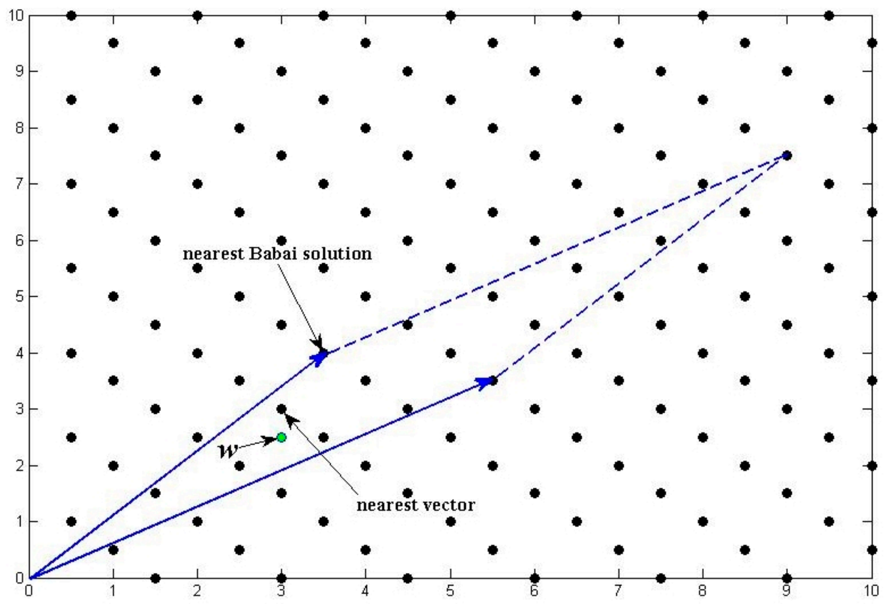
$$\min_{\mathbf{z} \in \mathbb{Z}^n} \|\hat{\mathbf{z}} - \mathbf{z}\|^2_{Q^{-1}_{\hat{\mathbf{z}}}} \tag{3}$$

where:

$$\begin{gathered} \hat{\mathbf{z}} = \mathbf{T}^{-1} \cdot \hat{\mathbf{a}} \\ \mathbf{z} = \mathbf{T}^{-1} \cdot \mathbf{a} \\ \mathbf{Q}^{-\mathbf{1}}_{\hat{\mathbf{z}}} = \mathbf{T}^T \cdot \mathbf{Q}^{-1}_{\hat{\mathbf{a}}} \cdot \mathbf{T} \end{gathered} \tag{4}$$

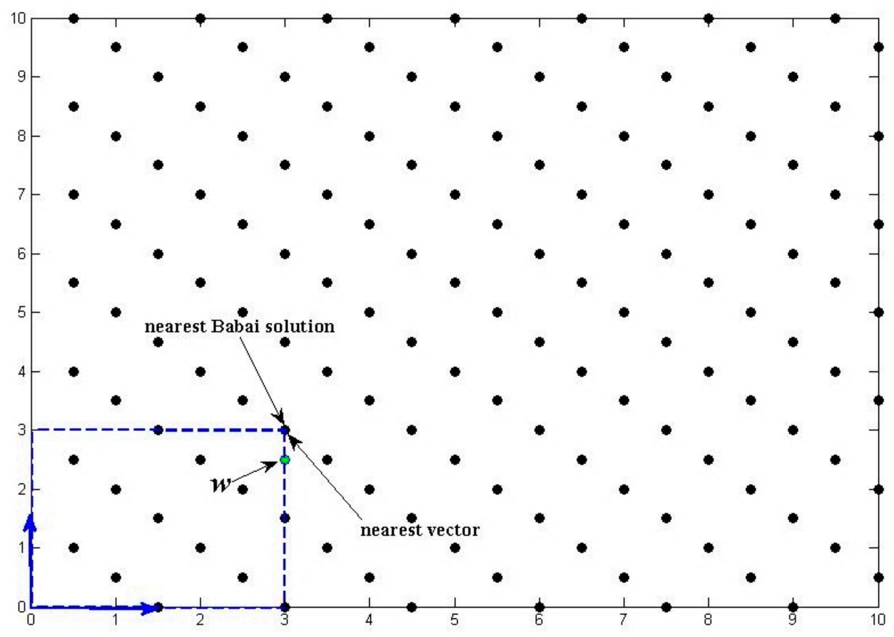
Assuming that we are able to find a proper unimodular matrix $\mathbf{T}$ that makes all the column vectors of $\mathbf{B}_o$ mutually orthogonal, the optimized solution of $\mathbf{z}$ can be obtained by rounding each entry in $\hat{\mathbf{z}}$ and the solution for $\mathbf{a}$ can be calculated accordingly, which is also known as the Babai's method [23]. However, such kind of matrix $\mathbf{T}$ cannot be found in general cases. As a result, the best way we can do is to find an approximate solution for matrix $\mathbf{T}$ that makes $\mathbf{B}_o$ as orthogonal as possible and the column vectors as short as possible, which is the so called "reduction" process.

Figure 1 shows the difference between a "bad" and a "good" reduced lattice basis and how they affect the search process. The basis vectors in Figure 1a are relatively long and have smaller angle, which will bring larger error when estimating the CVP solution for real value vector $\mathbf{w}$. On the contrary, the basis vectors in Figure 1b are mutually orthogonal, enabling the Babai's method to find the solution instantly.



(**a**)

Figure 1. *Cont.*



(**b**)

Figure 1. Illustration of "Bad" and "Good" lattice basis.

2.1. The LLL Reduction Algorithm

The two primary goals of reduction process are:

- To make the column vectors in $\mathbf{B}_o$ as mutually orthogonal as possible. As mentioned above, if the vectors are mutually orthogonal, then a simple rounding process will solve the ILS problem. Thus, the orthogonality of the vectors actually defines shape of search space, which will have significant influence on search efficiency;
- To minimize the length of vectors $\mathbf{b}_i$. Note that $\prod_{i=1}^{m} \|\mathbf{b}_i\|$ gives an upper bound of the volume of searching space, which means minimizing the vector length will shrinking the search space.

With the two goals, several reduction algorithm were proposed in the last decades and among those was the most famous LLL reduction algorithm proposed by Lenstra et al. [14].

The LLL reduction algorithm is consisted of two parts: size reduction and vector swap. The size reduction uses Gram-Schmidt orthogonalization. Let $\mathbf{B} = (\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3 \cdots \mathbf{b}_m)$ be a basis of lattice L and a Gram-Schmidt basis $\mathbf{B}^*$ can be described as:

$$\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^* \tag{5}$$

where:

$$\mu_{i,j} = \frac{\left\langle \mathbf{b}_i, \mathbf{b}_j^* \right\rangle}{\left\langle \mathbf{b}_j^*, \mathbf{b}_j^* \right\rangle} \tag{6}$$

Definition 1. *Given a basis $\mathbf{B} \in \mathbb{R}^{m \times m}$ of lattice L and its Gram-Schmidt orthogonal matrix $\mathbf{B}^*$, $\mathbf{B}$ is LLL reduced if it satisfies the following two conditions:*

$$\left|\mu_{i,j}\right| \leq \frac{1}{2} \text{ for } 1 \leq j < i \leq m \tag{7}$$

$$\delta\|\mathbf{b}_{i-1}^*\|^2 \leq \|\mathbf{b}_i^* + \mu_{i,i-1}\mathbf{b}_{i-1}^*\|^2 \text{ for } 1 < i \leq m \tag{8}$$

where parameter δ satisfies $\frac{1}{4} < \delta < 1$.

The LLL reduction algorithm starts with setting $\mathbf{b}_1^* = \mathbf{b}_1$ and then $\mathbf{b}_i$ is replaced by $\mathbf{b}_i - \lceil \mu_{i,j} \rfloor_{round} \mathbf{b}_j$ if $|\mu_{i,j}| \leq \frac{1}{2}$ for $1 \leq j < i \leq m$ to satisfy the size condition (7). If Lovász condition (8) is violated for $1 < i \leq m$, then the column vectors $\mathbf{b}_{i-1}$ and $\mathbf{b}_i$ are swapped and the size reduction process will go back to $\mathbf{b}_{i-1}$ again. After as much as $O(n^2 \log B)$ iterations [14], when the Lovász condition is satisfied between $\mathbf{b}_{m-1}$ and $\mathbf{b}_m$, the LLL reduction is done.

As we can see from the process of LLL reduction algorithm shown in Algorithm 1, the size reduction process adjusts the angle of $\mathbf{b}_i$ by rounded Gram-Schmidt orthogonalization and the length of $\mathbf{b}_i$ is reduced at the same time.

Algorithm 1: The LLL reduction algorithm

```
Set $\mathbf{b}_1^* = \mathbf{b}_1$
Set $i = 2$
For $i \leq m$
  For j = 1 to $i - 1$
    Set $\mathbf{b}_i = \mathbf{b}_i - \lceil \mu_{i,j} \rfloor_{round} \mathbf{b}_j$                (Size reduction)
  End
  If $\delta\|\mathbf{b}_{i-1}^*\|^2 > \|\mathbf{b}_i^* + \mu_{i-1,i}\mathbf{b}_i^*\|^2$ then         (Lovász condition)
    Swap $(\mathbf{b}_{i-1}, \mathbf{b}_i)$                           (Swap process)
    Set $i = \max(i - 1, 2)$
  Else
    Set $i = i + 1$
End
End
```

Clearly, the performance of size reduction process is strongly dependent on the order of the column vectors. Putting shorter vectors ahead will help improving the performance because shorter vectors kind of "push" the vectors after them to a more orthogonal angle in order to satisfy the size condition. Ideally, if we have:

$$\|\mathbf{b}_1\| \leq \|\mathbf{b}_2\| \leq \|\mathbf{b}_3\| \cdots \leq \|\mathbf{b}_m\| \tag{9}$$

and then the lattice can be most reduced. But unfortunately, there is still no algorithm can achieve (9) in polynomial time. As a result, Lovász set the condition to (8), where δ is usually set to 0.75 [14], in order to make the algorithm finish within polynomial time.

2.2. The LLL Reduction with Pivoted Reflection

As can be seen from Algorithm 1, the loop indicator i decreases only when the swaps take place, which means the number of iterations is highly dependent on the number of swaps. Hence, if the column vectors $\mathbf{b}_i$ are in ascending order or at least as ascending ordered as possible before the LLL reduction is executed, the number of swaps as well as the reduction time will surely decline.

Motivated by Chang's MLAMBDA [11] which utilizes the symmetric pivoting strategy to improve the efficiency of the reduction process and Wübben's MMSE Sorted QR decomposition [24] which extends the V-BLAST algorithm, we introduce the pivoted reflection strategy into LLL algorithm to pre-sort the column vectors.

As the Gram-Schmidt orthogonalization process in the original LLL algorithm is not an isometric process, the pivoting of the vectors has limited influence on the reduction time. Therefore, we chose the Householder transformation which is an elementary reflection transformation proposed by Turnbull and Aitken in 1932. The typical usage of the Householder reflection is QR decomposition.

Given a none-zero vector $\mathbf{b} \in \mathbb{R}^n$, one can construct vector $\mathbf{u} = \mathbf{b} + \rho\mathbf{e}$ with $\mathbf{e} = \left[1, \underbrace{0 \cdots 0}_{n-1 zeros}\right]^T$ and $\rho = sign(b_1) \cdot \|\mathbf{b}\|$. Then the transformation matrix can be constructed as:

$$\mathbf{H} = \mathbf{I} - 2\frac{\mathbf{u} \cdot \mathbf{u}^T}{\|\mathbf{u}\|^2} \tag{10}$$

Assuming $i - 1$ vectors of $\mathbf{B}$ have been transformed and $\mathbf{B}_i$ is the submatrix:

$$\mathbf{B}_i = \begin{bmatrix} b_{i,i} & \cdots & b_{1,m} \\ \vdots & \ddots & \vdots \\ b_{m,1} & \cdots & b_{m,m} \end{bmatrix} \tag{11}$$

then the ith transformation matrix can be calculated as:

$$\mathbf{H}_i = \begin{bmatrix} \mathbf{I}_{i-1} & 0 \\ 0 & \mathbf{H}_{\mathbf{B}_i} \end{bmatrix} \tag{12}$$

where $\mathbf{H}_{\mathbf{B}_i}$ is the transformation matrix for submatrix $\mathbf{B}_i$ according to (10). Then we get:

$$\mathbf{R} = \mathbf{H}_{m-1}\mathbf{H}_{m-2} \cdots \mathbf{H}_1\mathbf{B} \text{ and } \mathbf{Q} = \mathbf{H}_1\mathbf{H}_2 \cdots \mathbf{H}_{m-1} \tag{13}$$

with $\mathbf{B} = \mathbf{Q} \cdot \mathbf{R}$, where $\mathbf{Q}$ is an orthogonal matrix and $\mathbf{R}$ is an upper triangle matrix.

In order to make the column vectors as ascending ordered as possible, before each Householder transformation the shortest vector in submatrix $\mathbf{B}_i = \left[\mathbf{b}_1^i, \mathbf{b}_2^i, \mathbf{b}_3^i \cdots \mathbf{b}_{m-i+1}^i\right]$ is moved ahead and the corresponding column vector in $\mathbf{B}$ is swapped with $\mathbf{b}_i$ accordingly. The whole LLL reduction with pivoted reflection is given in Algorithm 2.

Algorithm 2: The LLL reduction with pivoted reflection

```
Set $\mathbf{R} = \mathbf{B}$ and $\mathbf{Q} = \mathbf{I}_{m \times m}$
For $i$ = 1 to m − 1                                   (Pivoted Householder Reflection)
  Find the shortest vector $\mathbf{b}_s^i$ in $\mathbf{B}_i$
  Swap($\mathbf{b}_i$, $\mathbf{b}_{i+s-1}$) of B
  Calculate $\mathbf{H}_i$ for B
  Set $\mathbf{R} = \mathbf{H}_i \cdot \mathbf{R}$ and $\mathbf{Q} = \mathbf{Q} \cdot \mathbf{H}_i$
End
Set $k$ = 2
While $k \leq m$
  For j = $k$-1 down to 1
  Set $\mathbf{r}_k = \mathbf{r}_k - \left[\frac{r_{j,k}}{r_{j,j}}\right]_{round} \cdot \mathbf{r}_j$          (Size reduction)
  End
  If $\delta \cdot r_{k-1,k-1}^2 > r_{k,k}^2 + r_{k-1,k}^2$ then          (Lovász condition)
    Swap ($\mathbf{r}_{k-1}$,$\mathbf{r}_k$)                              (Swap process)
    Calculate $\mathbf{H}_{k-1}$ for R
    Set $\mathbf{R} = \mathbf{H}_{k-1} \cdot \mathbf{R}$ and $\mathbf{Q} = \mathbf{Q} \cdot \mathbf{H}_{k-1}$
  Set $k$ = max($k$ − 1, 2)
  Else
    Set $k$ = $k$ + 1
End
End
```

Furthermore, the algorithm can be more efficient by applying Givens transformation to the rotations in the swap process instead of Householder reflection, because only $b_{i,i-1}$ needs to be transformed to zero when swapping $\mathbf{b}_{i-1}$ and $\mathbf{b}_i$. It should also be emphasized that the pivoted reflection strategy does not always sort the vectors in perfect ascending order, because the reflection on $\mathbf{b}_i$ also transforms the vectors after it, which may create shorter vector. But as can be seen in the simulations in Section 4, the pivoted reflection is proved to be very effective in minimizing the number of vector swaps in LLL algorithm in many cases of interest.

3. N-Dimensional Expansion of LLL Reduction

As discussed at the end of Section 2.1, the reduction quality and the improvement of search space are both highly dependent on the order of the basis vectors. We propose the n-LLL reduction algorithm, which inherits the basic outline of the LLL reduction algorithm and strengths the constraint of the order of basis vectors.

3.1. The N-Dimensional LLL Reduction Algorithm

Look back at the two conditions of LLL reduction again:

$$\left|\mu_{i,j}\right| \le \frac{1}{2},\ \delta\|\mathbf{b}^*_{i-1}\|^2 \le \|\mathbf{b}^*_i + \mu_{i,i-1}\mathbf{b}^*_{i-1}\|^2$$

and the reduction process can be described in another way as: applying a series of Gaussian reductions in the 2-dimensional lattice spanned by a Lovász condition optimized vector pair $\mathbf{b}_{i-1}$ and $\mathbf{b}_i$.

However, the Lovász condition here only focuses on local optimization within two neighbor vectors which ignores the global optimization. In order to improve the effect of the optimization to enhance the ordering constraint, we extend the 2-dimensional condition to n-dimension. Like the LLL reduction defined in Definition 1, the n-dimensional LLL reduction is defined as:

Definition 2. *Given a basis* $\mathbf{B} \in \mathbb{R}^{m\times m}$ *of Lattice L and its Gram-Schmidt orthogonal matrix* $\mathbf{B}^*$, $\mathbf{B}$ *is n-LLL reduced* ($2 \le n < m$) *if it satisfy the following two conditions:*

$$\left|\mu_{i,j}\right| \le \frac{1}{2} \text{ for } 1 \le j < i \le m \tag{14}$$

$$\delta\|\mathbf{b}^*_{i-1}\|^2 \le \min\left(\|\mathbf{b}^*_i + \mu_{i,i-1}\mathbf{b}^*_{i-1}\|^2, \cdots, \|\mathbf{b}^*_{i+n-2} + \sum_{j=i-1}^{\min(i+n-3,m)} \mu_{i+n-2,j}\mathbf{b}^*_j\|^2 \right) \text{ for } 1 < i \le m \tag{15}$$

where parameter δ *satisfies:* $\frac{1}{4} < \delta < 1$.

Condition (15) can also be rewritten as:

$$\delta\|\mathbf{b}^*_{i-1}\|^2 \le \lambda_1\left(\left[\mathbf{b}^i_1, \mathbf{b}^i_2, \mathbf{b}^i_3 \cdots \mathbf{b}^i_{\min(n-1,m-i+1)}\right]\right)^2 \tag{16}$$

where $\mathbf{b}^i_k$ is the column vector in submatrix $\mathbf{B}_i$ and $\lambda_1(\mathbf{B})$ implies the length of the shortest none-zero vector in $\mathbf{B}$.

Both condition (15) and (16) ensure that $\mathbf{b}_{i-1}$ is the optimized choice in the following n vectors, which brings stronger constraint to the reduced basis than the original Lovász condition. The n-LLL reduction becomes LLL reduction when $n = 2$.

Note that the extend Lovász condition in the n-LLL reduction is similar to Block Korkine-Zolotarev (BKZ) [21] reduction that applies Korkine-Zolotarev (KZ) reduction within a k-block. This paper extends the original Lovász condition instead of introducing KZ reduction basis.

As the constraint in n-LLL is stronger than in the original one, the increase of reduction time is predictable. Thus, pivoted reflection mentioned in Section 2.2 plays a vital role in controlling the total reduction time of the algorithm. So, we fuse QR decomposition and reduction together by deeply coupling the pivoted reflection and n-LLL reduction process.

One possible algorithm to achieve n-LLL reduction is shown in Algorithm 3:

Algorithm 3: The n-LLL reduction with pivoted reflection

```
Set $\mathbf{R} = \mathbf{B}$ and $\mathbf{Q} = \mathbf{I}_{m\times m}$
(Move the shortest vector to $\mathbf{r}_1$)
Find the shortest vector $\mathbf{r}_s$ in $\mathbf{R}$
Swap($\mathbf{r}_i$,$\mathbf{r}_s$) of $\mathbf{R}$
Calculate $\mathbf{H}_1$ for $\mathbf{R}$
Set $\mathbf{R} = \mathbf{H}_1 \cdot \mathbf{R}$ and $\mathbf{Q} = \mathbf{Q} \cdot \mathbf{H}_1$
Set $i = 2$
While $i \leq m$                          (Pivoted reflection and reduction process)
  Find the shortest vector $\mathbf{r}_s^i$ in $\mathbf{R}_i$
  Swap($\mathbf{r}_i$, $\mathbf{r}_{i+s-1}$) of $\mathbf{R}$
  Set temp = $i$
  For $j = i - 1$ down to 1
    Set $\mathbf{r}_k = \mathbf{r}_i - \left[\frac{r_{j,i}}{r_{j,j}}\right]_{round} \cdot \mathbf{r}_j$          (Size reduction)
    If $\delta \cdot r_{j,j}^2 > \|b_{temp-j+1}^j\|^2$ and $i - j < n$ then     (Extended Lovász condition)
      Swap ($\mathbf{r}_j$, $\mathbf{r}_{temp}$)                          (Swap process)
      Set temp = $j$
    End
  End
  Calculate $\mathbf{H}_{temp}$ for $\mathbf{R}$
  Set $\mathbf{R} = \mathbf{H}_{temp} \cdot \mathbf{R}$ and $\mathbf{Q} = \mathbf{Q} \cdot \mathbf{H}_{temp}$
  If $i$ ! = temp then
    $i$ = temp
  Else
    Set $i = i + 1$
  End
End
```

The algorithm above uses the "Pivoting-Reduction-Reflection" strategy for each vector of **B** instead of "Pivoting-Reflection and Reduction" strategy of Algorithm 2. The pivoting process in the fusion strategy gives better result as it takes the influence of the rotations in vector swap into consideration. Note that vector $\mathbf{r}_i$ may have less than $m - i$ zero entries because the vector pivoted to $\mathbf{r}_i$ has not been reflected yet. However, as the Householder reflection is an isometry, it will not be a problem in the swap process. The vector is then reflected after swapped to a proper place. The whole algorithm is actually executing the pivoted QR decomposition and n-LLL reduction in parallel, making it more efficient.

3.2. Analysis

3.2.1. The Performance

The n-LLL reduction algorithm strengthens the Lovász condition, which will certainly improve the quality of reduced basis. In this section, the performance improvement between the n-LLL and LLL reduction is detailed analyzed.

In order to compare the performance of reduction algorithm, we need to introduce measures to evaluate the reduction quality. As the size reduction processes of the two algorithms are the same, we only need to compare the orthogonality of the reduced basis according to the two goals mentioned in Section 2. One of the mostly used way to measure the orthogonality defects is imported from the Hadamard inequality:

Theorem 1. *(Hadamard Inequality): Given a lattice L and one of its basis* $\mathbf{B} \in \mathbb{R}^{m\times m}$*, we have:*

$$\det L = \det(\mathbf{B}) = \prod_{i=1}^{m} \|\mathbf{b}_i^*\| \le \prod_{i=1}^{m} \|\mathbf{b}_i\| \tag{17}$$

and it becomes an equation if and only if all the vectors are mutually orthogonal. Then, the orthogonality defect factor can be defined as:

$$OD(L(B)) = \frac{\prod\limits_{i=1}^{m} \|\mathbf{b}_i\|}{\det L} \tag{18}$$

with $1 \le O(L(B))$ *[25].*

According to (15), we have:

$$\delta\|\mathbf{b}_{i-1}^*\|^2 \le \|\mathbf{b}_i^*\|^2 + \mu_{i,i-1}^2\|\mathbf{b}_{i-1}^*\|^2 \tag{19}$$

$$\delta\|\mathbf{b}_{i-1}^*\|^2 \le \|\mathbf{b}_{i+1}^*\|^2 + \mu_{i+1,i}^2\|\mathbf{b}_i^*\|^2 + \mu_{i+1,i-1}^2\|\mathbf{b}_{i-1}^*\|^2 \tag{20}$$

$$\cdots\cdots$$

$$\delta\|\mathbf{b}_{i-1}^*\|^2 \le \|\mathbf{b}_{i+n-2}^*\|^2 + \sum_{j=i-1}^{\min(i+n-3,m)} \mu_{i+n-2,j}^2\|\mathbf{b}_j^*\|^2 \tag{21}$$

and the size reduction process makes sure that $\left|\mu_{i,j}\right| \le \frac{1}{2}$. As a result, (19) can be rewritten as:

$$\|\mathbf{b}_i^*\|^2 \ge \left(\delta - \frac{1}{4}\right)\|\mathbf{b}_{i-1}^*\|^2 \tag{22}$$

which means:

$$\|\mathbf{b}_{i+1}^*\|^2 \ge \left(\delta - \frac{1}{4}\right)\|\mathbf{b}_i^*\|^2 \tag{23}$$

By combining (23) and (20), we can get:

$$\|\mathbf{b}_{i+1}^*\|^2 \ge \frac{\left(\delta - \frac{1}{4}\right)^2\|\mathbf{b}_{i-1}^*\|^2}{\delta} \tag{24}$$

Repeatedly:

$$\begin{cases} \|\mathbf{b}_j^*\|^2 \le k^{i-j}\|\mathbf{b}_i^*\|^2 \\ k = \frac{\delta\cdot\delta^{\frac{1}{1-n}}}{\left(\delta-\frac{1}{4}\right)} \end{cases}, \frac{i-j}{n-1} \in \mathbb{Z} \tag{25}$$

Furthermore, by substituting (25) into the Gram-Schmidt orthogonalization (5), we get:

$$\begin{aligned} \|\mathbf{b}_i\|^2 &= \|\mathbf{b}_i^*\|^2 + \sum_{j=1}^{i-1}\mu_{i,j}^2\|\mathbf{b}_j^*\|^2 \le \|\mathbf{b}_i^*\|^2 + \frac{1}{4}\sum_{j=1}^{i-1}\|\mathbf{b}_j^*\|^2 \\ &\le \frac{3}{4}\|\mathbf{b}_i^*\|^2 + \frac{\|\mathbf{b}_i^*\|^2}{4}\sum_{j=0}^{\lfloor\frac{i-1}{n-1}\rfloor} k_n^{(n-1)j}\sum_{l=1}^{n-1} k_l^{l-1} \\ &\le \frac{3}{4}\|\mathbf{b}_i^*\|^2 + \frac{\|\mathbf{b}_i^*\|^2}{4}\sum_{j=0}^{\frac{i}{n-1}-1} k_n^{(n-1)j}\frac{k_n-1}{\delta k_2-1} \\ &= \frac{3}{4}\|\mathbf{b}_i^*\|^2 + \frac{\|\mathbf{b}_i^*\|^2}{4}\frac{k_n^i-1}{k_n^{n-1}-1}\frac{k_n-1}{\delta k_2-1} \end{aligned} \tag{26}$$

where $\lfloor a \rfloor$ denotes the integer no larger than a. By considering $1 < \delta k_2 < k_n \leq k_2 \leq k_n^{n-1}$, (26) can be further rewritten to:

$$\|\mathbf{b}_i\|^2 \leq \frac{k_n^i}{\delta k_2 - 1}\|\mathbf{b}_i^*\|^2 \tag{27}$$

Then the orthogonality defect factor is calculated:

$$OD(L) = \frac{\prod\limits_{i=1}^{m}\|\mathbf{b}_i\|}{\det L} \leq \sqrt{\frac{\prod\limits_{i=1}^{m}\frac{k_n^i}{\delta k_2 - 1}\|\mathbf{b}_i^*\|^2}{\prod\limits_{i=1}^{m}\|\mathbf{b}_i^*\|^2}} = \frac{k_n^{\frac{m(m+1)}{4}}}{(\delta k_2 - 1)^{\frac{m}{2}}} \tag{28}$$

It can be seen clearly from (28) and (25) that the upper bound of the orthogonality defect factor declines as the parameter n increasing, which proves that the reduction quality of n-LLL reduction algorithm with $n > 2$ is better than the original LLL reduction (which is equivalent to 2-LLL).

It should be emphasized that the orthogonality does not directly affect the search time according to [26,27]. However, as the two algorithms compared through the orthogonality defect factor share the same size reduction constraint, the difference of the factor actually measures the performance of vector ordering, or in another way measures the efficiency of size reduction. So, the conclusion of (28) still holds in numerical simulations in Section 4.

3.2.2. The Complexity

As can be seen in Algorithm 3, the loop indicator i goes back to *temp* after each vector swap, where *temp* indicates the final position that $\mathbf{b}_i$ is placed. Therefore, with *temp* $< i$, it is hard to determine that whether the algorithm will converge. Here we give the proof that the n-LLL reduction algorithm will converge in polynomial time.

Let:

$$L_i = \left\{ \sum_{k=1}^{i} a_k \mathbf{b}_k | a_k \in \mathbb{Z} \right\} \tag{29}$$

be a sub-lattice of lattice L and we define:

$$d_i = \prod_{k=1}^{i}\|\mathbf{b}_k\|^2 \tag{30}$$

$$D_i = \prod_{i=1}^{m} d_i \tag{31}$$

where we have $\det(L_i)^2 = d_i$.

In the n-LLL algorithm, the value of D_i changes only when the vector swap is executed. To be more accurate, only the elements from d_α to $d_{\beta-1}$ in D_i will change when swapping $\mathbf{b}_\alpha$ and $\mathbf{b}_\beta$. And the swap happens only when the extended Lovász condition is violated, which means:

$$\|\mathbf{b}_\beta^*\|^2 < \delta\|\mathbf{b}_\alpha^*\|^2 - \sum_{j=\alpha}^{\beta-1}\|\mu_{\beta,j}\mathbf{b}_j^*\|^2 \leq \delta\|\mathbf{b}_\alpha^*\|^2 \tag{32}$$

Therefore, the old d_ε^{old} will change to d_ε^{new} $(\alpha \leq \varepsilon < \beta)$:

$$\begin{aligned} d_\varepsilon^{new} &= \|b_1^*\|^2 \cdot \|b_2^*\|^2 \cdots \|b_\beta^*\|^2 \cdots \|b_\varepsilon^*\|^2 \\ &= \|b_1^*\|^2 \cdot \|b_2^*\|^2 \cdots \|b_\alpha^*\|^2 \cdots \|b_\varepsilon^*\|^2 \cdot \frac{\|b_\beta^*\|^2}{\|b_\alpha^*\|^2} \\ &\leq d_\varepsilon^{old} \cdot \delta \end{aligned} \tag{33}$$

and the maximum change of D_i by one swap is δ^{n-1}.

According to the Hermite principle, for a lattice $L \subset \mathbb{Z}^m$ we can have:

$$\min_{\mathbf{0} \neq \mathbf{v} \in L_i} \|\mathbf{v}\| \leq \sqrt{i} \cdot \det(L_i)^2 \tag{34}$$

which also means:

$$D_m = \prod_{i=1}^{m} d_i = \prod_{i=1}^{m} \det(L_i)^2 \geq \prod_{i=1}^{m} i^{-i} = (m!)^{-m} \geq m^{-m^2} \tag{35}$$

As a result, the value of D_m has a certain lower bound, which means only finite number of swaps are executed during the whole algorithm. Assuming that $D_m = D_{start}$ at the beginning of the algorithm and $D_m = D_{end}$ at the end, we have:

$$m^{-m^2} \leq D_{end} \leq \left(\delta^{n-1}\right)^N D_{start} \tag{36}$$

where $\boldsymbol{N}$ denotes the total number of vector swaps. Considering that $\delta < 1$, (36) can be rewritten as:

$$N = \frac{O\left(m^2 \log m + \log D_{end}\right)}{n-1} \tag{37}$$

and we also have:

$$D_{end} = \prod_{i=1}^{m} \|\mathbf{b}_i^*\|^{2(m+1-i)} \leq \prod_{i=1}^{m} \|\mathbf{b}_i\|^{2(m+1-i)} \leq (\max\|\mathbf{b}_i\|)^{m(m+1)} = B^{m(m+1)} \tag{38}$$

where B denotes the longest vector in $\mathbf{B}$.

Therefore, the total number swaps in the algorithm is:

$$N = O\left(n^{-1} \cdot m^2 \log B\right) \tag{39}$$

The fact is that the number of swaps declines as parameter n increases. However, the actual calculation effort is related to the number of loops. As the loop indicator i goes back as much as n steps after each vector swap instead of 1 step in the original LLL reduction algorithm which makes the maximum number of loops $n \cdot N + m$, which means each swap takes $O(n \cdot \log n)$ basic steps to satisfy extended Lovász condition in the n dimensional lattice space. Thus, $O\left(\log n \cdot m^3 \log B\right)$ basic steps are needed just to check all the vectors. Obviously, given an m rank basis, the calculation effort increases along with the increase of parameter n and the total reduction time remains polynomial for all $n \geq 2$.

4. Experiments and Results

4.1. Measures of Reduction Quality

In Section 3.2, we introduced the orthogonality defect factor to compare the n-LLL and the original LLL reduction algorithm. And we also mentioned that the orthogonality defect factor only measures the orthogonal quality of the two algorithms. In this section, we introduce a more practical measures, which is easier to calculate through the reduced matrix: the condition number [10].

The searching region which contains the nearest solution of ILS problem (3) can be written as:

$$(\hat{\mathbf{z}} - \mathbf{z})^T Q_{\hat{\mathbf{z}}}^{-1} (\hat{\mathbf{z}} - \mathbf{z}) < \chi^2 \tag{40}$$

and shape of this hyper-ellipsoid is determined by the ratio of the major and minor axes, which can be described as:

$$elongation = \sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}} = \sqrt{\kappa\left(Q_{\hat{\mathbf{z}}}\right)} \tag{41}$$

where $\kappa(Q_{\hat{\mathbf{z}}})$ is defined as the condition number of $Q_{\hat{\mathbf{z}}}$, $\lambda_{\max}$ and $\lambda_{\min}$ are the maximum and minimum eigenvalues of $Q_{\hat{\mathbf{z}}}$.

As a matter of fact, the condition number measures the elongation of the searching hyper-ellipsoid. Clearly, a lower condition number leads to a more sphere like search region and that will make the search more efficient and fast.

In order to illustrate the influence of condition number on the search time, Figure 2 shows a red ellipsoid defined by the original matrix and a reduced ellipsoid. The initial value for the search is often acquired by using Babai's method [23]. Supposing that we use the sphere search, then the minimum searching radius is the major axe of the hyper-ellipsoid. Clearly, by reducing it to the green ellipsoid showed in Figure 2, the size of the searching sphere is significantly shrunk down. Therefore, the condition number in some way reflects the search space for sphere searching method, which can be taken as an effective measure of reduction quality.

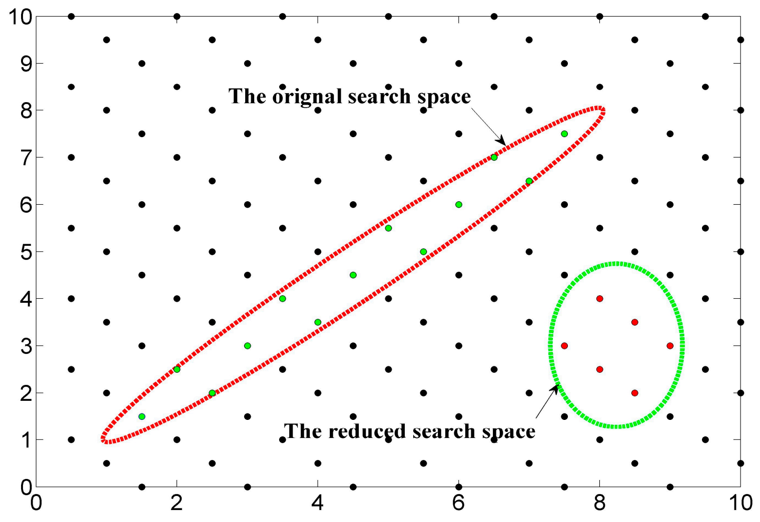


Figure 2. The effect of lattice reduction on the searching space.

4.2. Experiment Design

To evaluate the performance of the reduction algorithm thoroughly, the simulation matrix **B** should be designed carefully. Therefore, two major parameter should be focused on in particularly: the dimension m and the condition number κ. We first generate matrix **B** with two settings and then the covariance matrix $Q_{\hat{\mathbf{z}}}$ is calculated accordingly. Note that the weight matrix **W** is set to **I** without losing generality, as it has been incorporated into the basis as $\mathbf{B} = \mathbf{W}^{1/2}\mathbf{B}'$. The two simulation settings are:

Case 1: The $m \times m$ coefficient matrix $\mathbf{B}_{orignal}$ is firstly generated randomly by using the standard Gaussian normal distribution and then decomposed into matrixes $\mathbf{U}{\cdot}\mathbf{S}{\cdot}\mathbf{V}$ with the singular value decomposition. To control the condition number of the covariance matrix $Q_{\hat{\mathbf{z}}}$, the singular value matrix **S** is replaced by a diagonal matrix $\mathbf{S}'$, where $s'_1 = 2^{\frac{-\kappa}{4}}$, $s'_m = 2^{\frac{\kappa}{4}}$ and other entries of $\mathbf{S}'$ are randomly distributed between s'_1 and s'_m. The matrix **B** is then reconstructed as $\mathbf{B} = \mathbf{U}{\cdot}\mathbf{S}'{\cdot}\mathbf{V}$ and the covariance matrix is calculated as $Q_{\hat{\mathbf{z}}} = \mathbf{B}^T{\cdot}\mathbf{B}$ accordingly. Therefore, the condition number κ_{orig} of $Q_{\hat{\mathbf{z}}}$ is set to 2^{κ} with $\kappa = 5, 6, 7 \cdots 16$. For typical RTK application with all GNSS constellations like GPS, GLONASS, Galileo and BDS, setting the dimension of coefficient between 10 and 30 and the condition numbers between 2^8 and 2^{16} will cover most of the situations [12].

The first case covers most of the general purpose matrixes. As we paid more attention to the GNSS carrier phase resolution application of the lattice reduction algorithm, the unique features of the real GNSS signal should be taken into consideration. In RTK applications, the integer candidates are estimated mainly by sequential integer least-squares method. As Teunissen reported in [28,29], the spectrum of conditional variances shows great discontinuity during the sequential search. And the most significant gap always shows up between the third and the forth conditional variances.

This phenomenon can be briefly explained as follows. When solving the single short-baseline RTK positioning equations with double-difference carrier phase measurement, $(m + 3)$ unknowns are involved: m double-difference ambiguities and a 3-dimensional baseline vector. Assuming that three out of the m double difference ambiguities are already resolved, the baseline vector is then solvable with the corresponding three observation equations. At this moment, the other $(m - 3)$ double-difference ambiguities can also be solved precisely with $(m - 3)$ remaining observation equations. The conclusion is thus reached that once 3 ambiguities (with high conditional variances) are known, the remaining ambiguities can be determined with a very high precision (which means lower conditional variances). Here we use Case 2 to generate matrixes with this discontinuity features.

Case 2: Let $\mathbf{B} = \mathbf{U}\cdot\mathbf{S}'\cdot\mathbf{V}$, where $\mathbf{U}$ and $\mathbf{V}$ are obtained through the singular value decomposition like in Case 1 and $\mathbf{S}'$ is set to $\mathbf{S}' = diag\left(\sqrt{200}, \sqrt{200}, \sqrt{200}, \sqrt{0.1}\cdots\sqrt{0.1}\right)$, mimicking this discontinuity in the spectrum of the conditional variances.

Note that Case 2 is unlike the simulation case 4 in [11]. Instead of controlling the shape of the spectrum of conditional variances (the diagonal matrix in $\mathrm{L^TDL}$ decomposition) directly, Case 2 here controls the distribution of eigenvalues, which makes the generated matrixes share the same condition number.

In addition, the parameter δ is set to 0.75 for all the simulations considering of generality.

4.3. Performance of N-Dimensional LLL Reduction

4.3.1. Reduction Quality

The condition number κ mentioned in Section 4.1 describes how the search area looks like. In order to compare the LLL reduction algorithm and the n-LLL reduction algorithm, we let $\mathbf{Q}_{orig} = \mathbf{B}_{orig}^T\cdot\mathbf{B}_{orig}$, $\mathbf{Q}_{L^3} = \mathbf{B}_{L^3}^T\cdot\mathbf{B}_{L^3}$ and $\mathbf{Q}_{nL^3} = \mathbf{B}_{nL^3}^T\cdot\mathbf{B}_{nL^3}$ be the original covariance matrix, the LLL reduced covariance matrix and the n-LLL reduced covariance matrix and let κ_{orig}, κ_{L^3}, κ_{nL^3} be the condition numbers accordingly. Therefore, the improvement of the searching area can be described as:

$$\begin{aligned} d\kappa_{o-L^3} &= \log_{10}\left(\frac{\kappa_{orig}}{\kappa_{L^3}}\right) \\ d\kappa_{o-nL^3} &= \log_{10}\left(\frac{\kappa_{orig}}{\kappa_{nL^3}}\right) \\ d\kappa_{L^3-nL^3} &= \log_{10}\left(\frac{\kappa_{L^3}}{\kappa_{nL^3}}\right) \end{aligned} \tag{42}$$

We run the two cases mentioned above with three variables: (i) m, the dimension of the original coefficient matrix from 10 to 30 with an interval of 2; (ii) κ_{orig}, the condition number of covariance matrix from 8 to 16 with an interval of 1; (iii) n, the parameter for n-LLL reduction algorithm from 2 to m with an interval of 2. And for each setting, we perform 1000 independent random simulations to evaluate the algorithm.

Firstly, we pick out a set of simulations where $\kappa_{orig} = 2^{12}$(for Case 1 only), $m = 20$ and $n = 4$ and the probability density function of the $d\kappa$ is plotted in Figure 3. It can be seen clearly in Figure 3a,b ($d\kappa_{o-L^3}$) and Figure 3c,d ($d\kappa_{o-nL^3}$) that both the LLL reduction algorithm and the n-LLL reduction algorithm are able to reduce the condition number significantly. Furthermore, Figure 3e,f shows the difference between the two algorithms $d\kappa_{L^3-nL^3}$, which proves the 4-LLL reduction algorithm has larger probability to perform better than the original LLL reduction. Statistically, among the 1000 independent runs of this particular simulation setting, 64.7% results of the 4-LLL is better than that of the LLL with a maximum improvement factor of 5.85. And the average improvement is about 39%. It should be pointed out that for all the simulation settings mentioned above, the conclusion showed in Figure 3 holds in general.

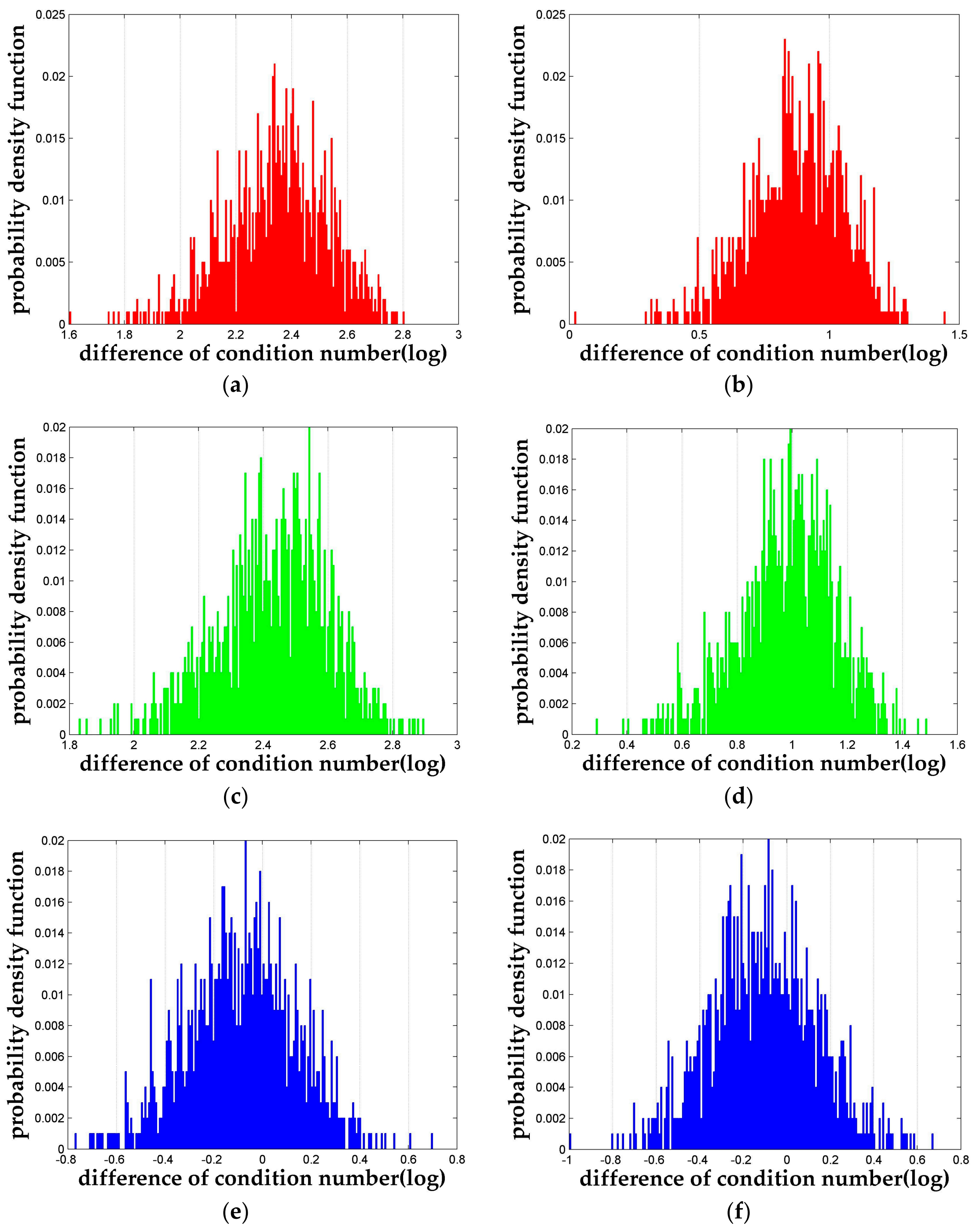
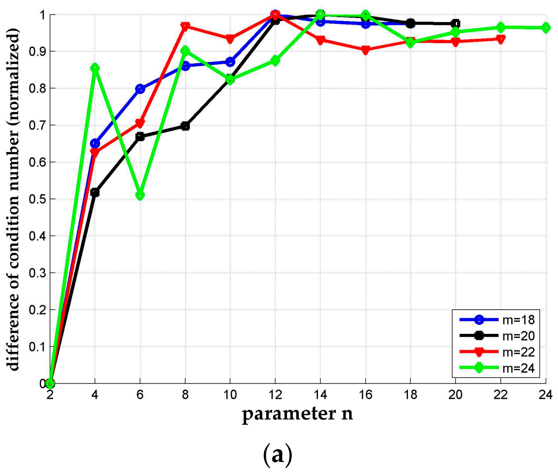


Figure 3. The probability density functions of the differences of condition numbers in logarithm to base 10. (**a**) $d\kappa$ between the original matrix and LLL reduced matrix for $\kappa_{orig} = 2^{12}$ of Case 1; (**b**) $d\kappa$ between the original matrix and LLL reduced matrix for Case 2; (**c**) $d\kappa$ between the original matrix and 4-LLL reduced matrix for $\kappa_{orig} = 2^{12}$ of Case 1; (**d**) $d\kappa$ between the original matrix and 4-LLL reduced matrix for Case 2; (**e**) $d\kappa$ between the LLL reduced matrix and 4-LLL reduced matrix for $\kappa_{orig} = 2^{12}$ of Case 1; (**f**) $d\kappa$ between the LLL reduced matrix and 4-LLL reduced matrix for Case 2.

In Section 3.2, inequality (28) implies that the lower bound of the reduction quality (the orthogonality of the coefficient matrix) improves as the parameter n increases. To prove this conclusion, we take $m = 20$ for example and let the parameter n vary from 4 to 20. The results shown in Figure 4 is not as expected and an upper limit shows up for the $d\kappa$ as the n increases. And the condition number of the output matrix changes no more after n reaches a certain number. The explanation for this phenomenon is that the column vectors in **B** cannot become "more ordered" to improve the orthogonality anymore at this point. However, Figure 4 still shows that $n = 2$ (the original LLL) is generally not this upper limit and there is space to improve reduction performance by increasing the parameter n.



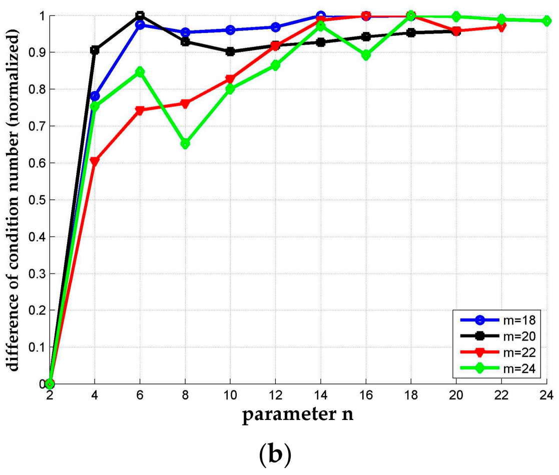
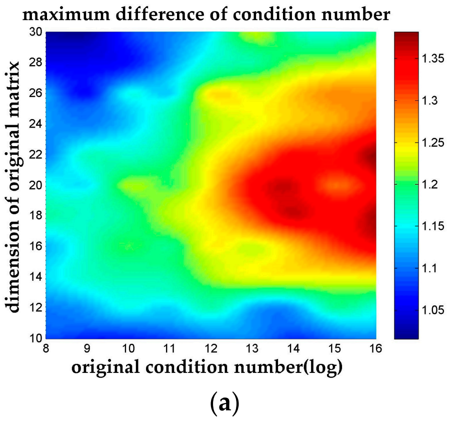


Figure 4. Normalized difference of condition numbers. The differences of condition number between n-LLL and the original LLL are normalized and plotted altogether with m range from 18 to 24. (**a**) The normalized difference of condition number for $\kappa_{orig} = 2^{14}$ of Case 1; (**b**) The normalized difference of condition number of Case 2.

This can be clearly seen in the thermodynamic diagram Figure 5, where the temperature denotes $d\kappa^{\max}_{L^3-nL^3} = \log_{10}\left(\frac{\kappa_{L3}}{\min(\kappa_{nL^3})}\right)$. The highest temperature shows up when the dimension of the coefficient matrix is around 20 with a relatively high κ_{orig} for Case 1, which indicates that the n-LLL reduction algorithm maximum its performance in that area. And for Case 2, the highest temperature also shows up around the dimension of 18 to 20. However, the difference between LLL and n-LLL becomes less significant when dealing with matrixes with higher dimension but lower condition number.



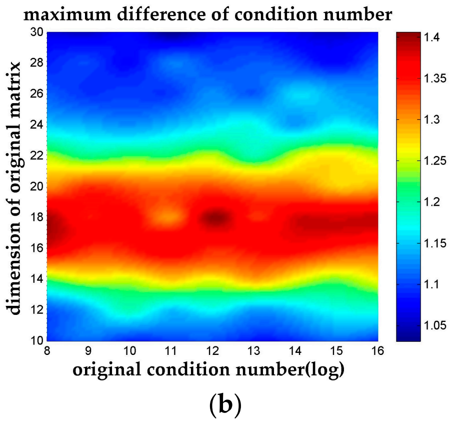
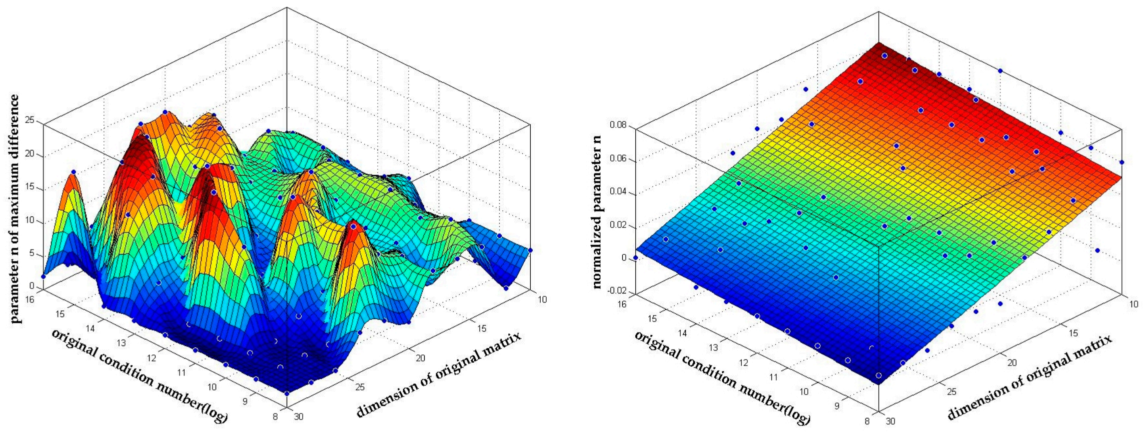


Figure 5. The maximum difference of condition numbers. The max value of the difference of condition number between n-LLL and LLL among all possible n parameter is plotted in the thermodynamic diagram. (**a**) The maximum difference of condition numbers for Case 1; (**b**) The maximum difference of condition numbers for Case 2.

As can be seen later in Section 4.3.2, setting a higher parameter n always results in a longer reduction time. Thus, finding an optimized n to balance the performance and the reduction time is important. We analyzed the simulation results deeply and found that parameter n is relevant to the square of the dimension. We give out the fit result based on our simulation in Figure 6 and (43). The goodness of the first order polynomial surface fitting in Figure 6b reaches 0.6851, which means it can be taken as an empirical reference formula within the simulation setting range.

$$n_{fit} = \left(0.0670 - 0.0027 \cdot m + 0.0014 \cdot \log_2 \kappa_{orig}\right) \cdot m^2 \tag{43}$$



(**a**) (**b**)

Figure 6. The parameter n with the best reduction quality. The corresponding parameter n of the max difference of condition numbers in Figure 5a. (**a**) The distribution of parameter n with the best reduction quality; (**b**) The 1st order fit of normalized parameter n/m^2.

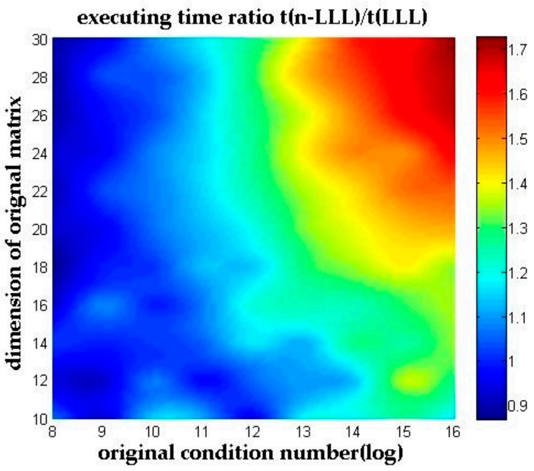
4.3.2. Complexity

In order to evaluate the effect of pivoted reflection, we compared the n-LLL reduction without pivoted reflection (PR), the n-LLL reduction with pivoted reflection and the original LLL reduction algorithm. The simulation is performed on an E5-2650 CPU which makes the executing time is relatively short. Thus, matrixes with high dimension and high condition number are chosen in this simulation to obtain more accurate time measurements. 1000 independent runs are performed for 30-dimensional matrixes with condition number range from 2^{10} to 2^{16}. And to evaluate the worst case of the n-LLL reduction algorithm, parameter n is fixed to 30 accordingly. Table 1 shows the simulation results, indicating that the pivoted reflection is very effective on improving the efficiency of the algorithm.

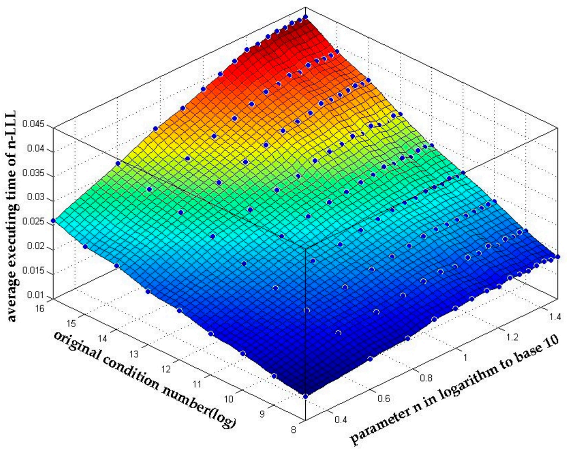
Table 1. The executing time of n-LLL reduction and LLL reduction.

Condition Number	n-LLL without PR	n-LLL with PR	Original LLL
2^{10}	536 ms	230 ms	217 ms
2^{11}	561 ms	251 ms	224 ms
2^{12}	595 ms	286 ms	222 ms
2^{13}	628 ms	316 ms	220 ms
2^{14}	651 ms	341 ms	222 ms
2^{15}	672 ms	369 ms	235 ms
2^{16}	714 ms	412 ms	239 ms

The conclusion is obvious that the pivoted reflection can reduce the reduction time by up to 57% and it is also clear that the n-LLL reduction algorithm with pivoted reflection causes no significant rise on the total reduction time compared with the original LLL reduction. And in the worst case scenario of all our simulations, the n-LLL reduction algorithm consumes no more than 1.72 times of the original LLL reduction time, which can also be observed in Figure 7a.



(**a**)



(**b**)

Figure 7. The executing time of n-LLL. (**a**) The executing time ratio between n-LLL and the original LLL algorithm; (**b**) The average executing time of n-LLL with parameter n presented in logarithm to base 10.

Take another look at the results in Figure 7a in a different angle where the parameter n is present in logarithm to base 10. The executing time in Figure 7b shows great linearity to the logarithm of parameter n, which meets the conclusion presented in Section 3.2.2 well.

Figure 8 shows the relationship between the matrix dimension, the inverse of parameter n and the average number of swaps in the reduction process. As we analyzed in Section 3.2.2, (39) shows that the number of swaps will actually decline with the increase of parameter n as long as both the dimension and the condition number of the original matrix is determined. And this conclusion can be clearly seen in Figure 8 that the number of swaps declines by n times.

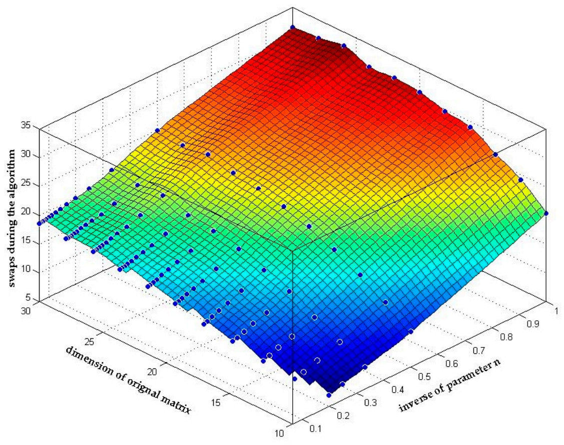


Figure 8. Number of swaps during the n-LLL algorithm.

5. Conclusions

This paper has presented a new kind of lattice reduction algorithm motivated by the original LLL reduction. In order to put more emphasize on the order of the basis vectors, the Lovász condition is further strengthened, expanding the 2-dimensional local optimization to n-dimensional global optimization. The pivoted reflection method based on the Householder transformation and QR decomposition is added to the algorithm in order to reduce the additional computational effort brought by the extra vector swaps.

By utilizing the Hadamard Inequality and the orthogonality defect factor, the relation between the n parameter of n-LLL and the reduction quality is illustrated. Analysis shows that with the increase of the parameter n, the basis vectors will become more orthogonal or at least its orthogonality defect factor will have a smaller lower bound. The complexity of the n-LLL algorithm is estimated afterwards, which shows the n-LLL algorithm consumes more elementary steps than the original LLL with $n > 2$. However, the analysis also proves that the n-LLL algorithm still remains a polynomial-time algorithm.

In the numerical simulation, two basic cases are covered to mimic the features of GNSS double-difference carrier-phase measurements. The simulation results show that the reduction quality of n-LLL algorithm is better than the original LLL reduction algorithm, especially for highly ill-conditioned matrixes. However, with the increase of the parameter n, a certain upper bound of the reduction quality of n-LLL is found during the simulation. At this certain point, the effect of permutation reaches its limit and further permutation will not significantly affect the reduction quality. In this case, a first-order surface fit is given to estimate this certain parameter n.

In order to evaluate the effect of pivoted reflection, the n-LLL reduction with and without pivoted reflection as well as the original LLL reduction are compared. The results clearly show the power of pivoted reflection, especially for matrix with relatively low condition number. And in the worst case scenario of all our simulation, the pivoted reflection ensures the n-LLL reduction algorithm consumes no more than 1.72 times of the original LLL reduction executing time.

Both the analysis and the simulation results have shown that the n-LLL algorithm is better than the original LLL reduction algorithm, especially for highly ill-conditioned matrixes. And the increase of the total reduction time is fairly acceptable, which makes the n-LLL a new practical algorithm for lattice basis reduction.

Acknowledgments: The work is supported by the National Key Research and Development Program of China (No. 2016YFB0502001) and National Natural Science Foundation of China (No. 61372110).

Author Contributions: Zhongliang Deng proposed the general idea of the n-LLL reduction and pointed out the importance of the order of basis vectors; Di Zhu designed the algorithm, added pivoted reflection into the algorithm, performed the analysis and simulation and wrote the paper; Lu Yin helped to enrich the simulation results and to revise the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Provenzano, J.P.; Masson, A.; Journo, S.; Laramas, C.; Damidaux, J.I.; Zwolska, F. Implementation of a european infrastructure for satellite navigation: From system design to advanced technology evaluation. *Int. J. Satell. Commun. Netw.* **2015**, *18*, 223–242. [CrossRef]
2. Dow, J.M.; Neilan, R.E.; Rizos, C. The international GNSS service in a changing landscape of global navigation satellite systems. *J. Geodesy* **2009**, *83*, 191–198. [CrossRef]
3. Gerke, M.; Przybilla, H. Accuracy analysis of photogrammetric UAV image blocks: Influence of onboard RTK-GNSS and cross flight patterns. *Photogramm. Fernerkund. Geoinf.* **2016**, *2016*, 17–30. [CrossRef]
4. Tahar, K.N.; Ahmad, A.; Wan, A.A.W.M.A.; Wan, M.N.W.M. Unmanned aerial vehicle photogrammetric results using different real time kinematic global positioning system approaches. In *Lecture Notes in Geoinformation & Cartography*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 123–134.

5. Woo, H.J.; Yoon, B.J.; Cho, B.G.; Kim, J.H. Research into navigation algorithm for unmanned ground vehicle using real time kinemtatic (RTK)-GPS. In Proceedings of the 2009 ICCAS-SICE, Fukuoka, Japan, 18–21 August 2009; pp. 2425–2428.
6. Hatch, R. *Instantaneous Ambiguity Resolution*; Springer: New York, NY, USA, 1991.
7. Forssell, B.; Martinneira, M.; Harrisz, R.A. Carrier phase ambiguity resolution in GNSS-2. In Proceedings of the 10th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS 1997), Kansas City, MO, USA, 16–19 September 1997; pp. 1727–1736.
8. Teunissen, P.J.G. The least-squares ambiguity decorrelation adjustment: A method for fast GPS integer ambiguity estimation. *J. Geodesy* **1995**, *70*, 65–82. [CrossRef]
9. Teunissen, P.J.G. Least-Squares Estimation of the Integer GPS Ambiguities. Invited lecture, Section IV "Theory and Methodology". In Proceedings of the General Meeting of the International Association of Geodesy, Beijing, China, August 1993.
10. Xu, P. Random simulation and gps decorrelation. *J. Geodesy* **2001**, *75*, 408–423. [CrossRef]
11. Chang, X.W.; Yang, X.; Zhou, T. MLAMBDA: A modified LAMBDA method for integer least-squares estimation. *J. Geodesy* **2005**, *79*, 552–565. [CrossRef]
12. Xu, P. Parallel cholesky-based reduction for the weighted integer least squares problem. *J. Geodesy* **2012**, *86*, 35–52. [CrossRef]
13. Hassibi, A.; Boyd, S. Integer parameter estimation in linear models with applications to GPS. *IEEE Trans. Signal Process.* **1998**, *46*, 2938–2952. [CrossRef]
14. Lenstra, A.K.; Lenstra, H.W.; Lovász, L. Factoring polynomials with rational coefficients. *Math. Ann.* **1982**, *261*, 515–534. [CrossRef]
15. Wen, J.; Chang, X.W. GfcLLL: A greedy selection-based approach for fixed-complexity LLL Reduction. *IEEE Commun. Lett.* **2017**, *21*, 1965–1968. [CrossRef]
16. Taherzadeh, M.; Mobasher, A.; Khandani, A.K. LLL reduction achieves the receive diversity in MIMO decoding. *IEEE Trans. Inf. Theory* **2007**, *53*, 4801–4805. [CrossRef]
17. Ying, H.G.; Cong, L.; Mow, W.H. Complex lattice reduction algorithm for low-complexity MIMO detection. *IEEE Trans. Signal Process.* **2006**, *5*, 2701–2710.
18. Grafarend, E.W. Mixed integer-real valued adjustment (IRA) problems: GPS initial cycle ambiguity resolution by means of the LLL algorithm. *GPS Solut.* **2000**, *4*, 31–44. [CrossRef]
19. Nguên, P.Q.; Stehlé, D. *Floating-Point LLL Revisited*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 215–233.
20. Schnorr, C.P.; Euchner, M. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Math. Program.* **1994**, *66*, 181–199. [CrossRef]
21. Schnorr, C.P. A hierarchy of polynomial time basis reduction algorithms. *Theor. Comput. Sci.* **1987**, *53*, 375–386. [CrossRef]
22. Fontein, F.; Schneider, M.; Wagner, U. PotLLL: A polynomial time version of LLL with deep insertions. *Des. Codes Cryptogr.* **2014**, *73*, 355–368. [CrossRef]
23. Babai, L. On Lovász' lattice reduction and the nearest lattice point problem. In Proceedings of the STACS 85 Symposium on Theoretical Aspects of Computer Science, Saarbrücken, Germany, 3–5 January 1985; pp. 13–20.
24. Wübben, D.; Bohnke, R.; KüHn, V.; Kammeyer, K.D. MMSE extension of V-BLAST based on sorted QR decomposition. In Proceedings of the Vehicular Technology Conference, Orlando, FL, USA, 6–9 October 2003; Volume 501, pp. 508–512.
25. Vetter, H.; Ponnampalam, V.; Sandell, M.; Hoeher, P.A. Fixed complexity LLL algorithm. *IEEE Trans. Signal Process.* **2009**, *57*, 1634–1637. [CrossRef]
26. Borno, M.A.; Chang, X.W.; Xie, X.H. On "decorrelation" in solving integer least-squares problems for ambiguity determination. *Emp. Surv. Rev.* **2014**, *46*, 37–49. [CrossRef]
27. Xu, P. Voronoi cells, probabilistic bounds, and hypothesis testing in mixed integer linear models. *IEEE Trans. Inf. Theory* **2006**, *52*, 3122–3138. [CrossRef]
28. Jonge, P.P.D.; Tiberius, C. On the spectrum of the GPS DD-ambiguities. In Proceedings of the 7th International Technical Meeting of the Satellite Division of the Institute of Navigation, Salt Lake City, UT, USA, 20–23 September 1994.
29. Teunissen, P.J.G.; Jonge, P.P.D.; Tiberius, C. The least-squares ambiguity decorrelation adjustment: Its performance on short GPS baselines and short observation spans. *J. Geodesy* **1997**, *71*, 589–602. [CrossRef]