An Extension of LLL Algorithm and its relationship to LNC performance

Lattice Reduction over Complex Plane
with application in Lattice Network Coding
I. Preliminaries
   a) Lattice
   b) LLL Algorithm and Closest Vector Problem

II. Complex Lattice ($\mathbb{Z}[i]$ and $\mathbb{Z}[\omega]$) and Complex LLL Algorithm

III. Lattice Network Coding
   a) Model and Operation
   b) Optimization and its Relation to LLL Algorithm

IV. Simulation Results

V. Acknowledgement and References
Notation Convention

• Throughout this presentation, we shall adapt following notations:
  • $N_0$: The (expected/average) power of Gaussian White Noise in dB;
  • $n_0$: Above in magnitude, i.e. $n_0 = 10^{\frac{N_0}{10}}$;
  • $snr$: Signal-to-Noise Ratio in magnitude, defined as $snr = \frac{p}{n_0}$, where $p$ is the expected power of the sending signal (in magnitude);
  • $SNR$: Above in dB, i.e. $SNR = 10 \cdot \log_{10} snr$. 
Lattice and LLL Algorithm
Preliminaries
Lattice

• Lattice generated by two real vectors $x_1$ and $x_2$:
  \[ L = \mathbb{Z}x_1 + \mathbb{Z}x_2 \]
  
  • as an analog to the vector space generated by $x_1$ and $x_2$: $\mathbb{R}x_1 + \mathbb{R}x_2$
Sublattice

- Sublattice generated by two real vectors $2x_1$ and $2x_2$:
  \[ L' = \mathbb{Z}(2x_1) + \mathbb{Z}(2x_2) \]
- as an analog to the vector space generated by $x_1$ and $x_2$: $\mathbb{R}x_1 + \mathbb{R}x_2$
Find: The shortest nonzero vector in $L$, i.e. the shortest vector $v = \sum_i a_i l_i$ with $a_i \in \mathbb{Z}$ not all zero.
Closest Vector Problem (CVP)

1) \( l_1 = (9,5), l_2 = (3,3) \)
   \(|l_1| > |l_2| \) and \( l_1 = 2 \cdot l_2 + (3,-1) \)
   \( l_1 \leftarrow (3,-1) \)

2) \( l_2 = (3,-1), l_1 = (3,3) \)
   \(|l_1| > |l_2| \) and \( l_1 = l_2 + (0,4) \)
   \( l_1 \leftarrow (0,4) \)

3) No improvement possible
   Stop
Gaussian Algorithm

Result:

\[
\begin{bmatrix}
9 & 5 \\
3 & 3
\end{bmatrix} \rightarrow \begin{bmatrix}
0 & 4 \\
3 & 1
\end{bmatrix}
\]

The first one is indeed shortest (non-zero) vector of the lattice;

Bonus:

The second one is the second shortest.
Extend to 3D Version

- Generalize the idea from Gaussian Algorithm:
- How about 3 vectors?

$$l_1 = (0, 4, 0)$$

$$l_2 = (3, -1, 0)$$

$$l_3 = (5, 6, 4)$$

$$\begin{bmatrix} 9 & 5 & 0 \\ 3 & 3 & 0 \\ 5 & 6 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 4 & 0 \\ 3 & 1 & 0 \\ 5 & 6 & 4 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 4 & 0 \\ 3 & 1 & 0 \\ 5 & 6 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 4 & 0 \\ 3 & 1 & 0 \\ 2 & 5 & 4 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 4 & 0 \\ 3 & 1 & 0 \\ 2 & 5 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 4 & 0 \\ 2 & 5 & 4 \\ 3 & 1 & 0 \end{bmatrix}$$

Reduce
Swap shorter ahead
Reduce
Swap shorter ahead: In terms of projection height on $l_1$
Extend to 3D Version

How about 3 vectors?

\[
\begin{bmatrix}
9 & 5 & 0 \\
3 & 3 & 0 \\
5 & 6 & 4
\end{bmatrix}
\rightarrow
\begin{bmatrix}
0 & 4 & 0 \\
3 & 1 & 0 \\
5 & 6 & 4
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & 4 & 0 \\
3 & 1 & 0 \\
5 & 6 & 4
\end{bmatrix}
\rightarrow
\begin{bmatrix}
0 & 4 & 0 \\
3 & 1 & 0 \\
2 & 5 & 4
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & 4 & 0 \\
3 & 1 & 0 \\
2 & 5 & 4
\end{bmatrix}
\rightarrow
\begin{bmatrix}
0 & 4 & 0 \\
2 & 1 & 4 \\
3 & 1 & 0
\end{bmatrix}
\]

Generalize the idea from Gaussian Algorithm:

Run Gaussian on first 2 vectors

Swap shorter ahead: In terms of projection height on \( l_1 \)

\[
\begin{bmatrix}
0 & 4 & 0 \\
2 & 5 & 4 \\
3 & 1 & 0
\end{bmatrix}
\rightarrow
\begin{bmatrix}
0 & 4 & 0 \\
2 & 1 & 4 \\
3 & 1 & 0
\end{bmatrix}
\]

Goes back, Run Gaussian

Try Reduce Again

No Improvement Possible, Reduce \( l_3 \) on \( l_1 \)
LLL Algorithm

- It cannot be proven that the LLL algorithm is polynomial-time.
- It can be with the modification below.

Fix a parameter $\alpha \in (\frac{1}{4}, 1)$. Swap when the projection height of $l_{k+1}$ is smaller than $\alpha$ times that of $l_k$.

LLL on $k+1$ vectors ($k \geq 1$)

- Run LLL on first $k$ real vectors
- Reduce $l_{k+1}$ on $l_k$
- Reduce $l_{k+1}$ on $l_{k-1}, l_{k-2}, \ldots, l_1$

Swap shorter ahead: In terms of projection height on the space spanned by $\{l_1, l_2, \ldots, l_{k-1}\}$

No Swap
Polynomial-time Complexity of LLL Algorithm

• It can be proven that LLL can finish within

\[
\text{Time} = O(n^2 \log B) \cdot \text{Time(swap)} = O(n^4 \log B)
\]

Times of “swapping”, where \( B \) is length of longest input vector, \( n \) is number of them.
Near Optimality of LLL

- It can also be proven that for a set of input \( \{l_1, l_2, \cdots, l_n\} \) after LLL:

\[
\|l_1\| \leq \beta^{\frac{n-1}{2}} \cdot \|l\|
\]

where \( \beta = \frac{4}{4\alpha - 1} \in \left(\frac{4}{3}, \infty\right) \) and \( l \) is the shortest vector.
Lattice over \( \mathbb{Z}[i] \) and \( \mathbb{Z}[\omega] \) and Complex LLL Algorithm

Complex Extension
Gaussian integer $\mathbb{Z}[i]$ and Eisenstein integer $\mathbb{Z}[^\omega]$

- **Ring of Gaussian integers:**
  \[ \mathbb{Z}[i] = \{a + bi | a, b \in \mathbb{Z}\} \]
  where \(i^2 = -1\)

- **Ring of Eisenstein integers:**
  \[ \mathbb{Z}[\omega] = \{a + b\omega | a, b \in \mathbb{Z}\} \]
  where \(\omega = e^{\frac{2\pi i}{3}} = -\frac{1}{2} + \frac{i\sqrt{3}}{2}\)
From lattice over \( \mathbb{Z} \) to \( \mathbb{Z}[i] \) and \( \mathbb{Z}[\omega] \)

<table>
<thead>
<tr>
<th>Lattice over ( \mathbb{Z} ) is of form: ( L = \mathbb{Z}x_1 + \mathbb{Z}x_2 + \cdots + \mathbb{Z}x_K ) for real vectors ( x_1, x_2, \ldots, x_K )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lattice over ( \mathbb{Z}[i] ) is of form: ( L = \mathbb{Z}[i]x_1 + \mathbb{Z}[i]x_2 + \cdots + \mathbb{Z}[i]x_K ) for complex vectors ( x_1, x_2, \ldots, x_K )</td>
</tr>
<tr>
<td>Lattice over ( \mathbb{Z}[\omega] ) is of form: ( L = \mathbb{Z}[\omega]x_1 + \mathbb{Z}[\omega]x_2 + \cdots + \mathbb{Z}[\omega]x_K ) for complex vectors ( x_1, x_2, \ldots, x_K )</td>
</tr>
</tbody>
</table>

To improve the performance of LLL algorithm, consider lattice over \( \mathbb{Z}[i] \) and \( \mathbb{Z}[\omega] \).
Reduce $l_K$ on $\mathbb{Z}[i]$ or $\mathbb{Z}[\omega]$

LLL on $k + 1$ vectors ($k > 1$)

Run LLL on first $k$ complex vectors

Reduce $l_{k+1}$ on $l_k$

Swap shorter ahead: In terms of projection height on the space spanned by $\{l_1, l_2, \cdots, l_{k-1}\}$

No Swap

Reducing $l_{k+1}$ on $l_{k-1}$, $l_{k-2}$, $\cdots$, $l_1$
Operation, Performance and Relation to LLL Algorithm

Lattice Network Coding
The Compute and Forward Model

Lattice Network Coding is a coding scheme based on multi-user physical-layer network coding, which is often abstracted by following model
The Compute and Forward Model

\[ y = \sum_{i=1}^{N} h_i x_i + z \]

\[ u = \sum_{i=1}^{N} a_i w_i \]
Operation - Encode

- Code words: $\omega_i \in (\Lambda/ \Lambda')^n$
- Encode input code words (a coset) to its coset representative.
- Example:

$$\Lambda = \mathbb{Z}[\omega] = \{a + b \cdot \left(-\frac{1}{2} + \frac{\sqrt{3}}{2}i\right)\}$$

$$\Lambda' = 2 \cdot \Lambda$$

$$W = (\mathbb{Z}[\omega]/2\mathbb{Z}[\omega])^n$$

$$= \{\text{codes} \}^n$$

$x_1, x_2, ..., x_N \in \mathbb{C}^n$
Operation - Decode

- **Conditional Decoding:**
  1. Coefficient Parameter (User Specified): 
     \[ \mathbf{a} = (a_1, a_2, \ldots, a_N) \]
  2. Augment Parameter: \( \alpha \in \mathbb{C} \) (to decrease error probability)
  3. Target decode output: 
     \[ \mathbf{u} = a_1 \mathbf{w}_1 + a_2 \mathbf{w}_2 + \cdots + a_N \mathbf{w}_N \]
Optimization
- Minimize Error Probability

- The error of this model highly depends on the choice of 
  \( a = (a_1, a_2, \ldots, a_N) \) and \( \alpha \).
- Best \( \alpha \) can be found according to \( a \).
- Best \( a \) should be chosen to minimize \( aMa^H \), i.e. to minimize \( \|aL\| \)

where \( L \) is a lower triangular matrix s.t. \( M = LL^H \)

**Given** \( a \), **we can find BEST** \( \alpha \) as follows: [Nazer-Gastpar’11]

\[
\alpha = \frac{a \cdot h^H \text{snr}}{\text{snr} \|h\|^2 + 1}
\]

The UBE of decoding error probability is [Tyler-Sun’13]

\[
P_{\text{error}} \approx K \left( \frac{\Lambda}{\Lambda'} \right) \exp \left( - \frac{d^2 \left( \frac{\Lambda}{\Lambda'} \right)}{4n_0 aMa^H} \right)
\]

where \( M = \text{snr}I_N - \frac{\text{snr}^2}{\text{snr} \|h\|^2 + 1} hh^H \), \( d \): length of shortest vectors in \( \frac{\Lambda}{\Lambda'} \), \( K \): number of these vectors.

Let \( L \) be a lower triangular matrix such that \( M = LL^H \).
Relation to LLL Algorithm  
- Solve a shortest vector problem

• To minimize decoding error probability, parameter vector $\mathbf{a} = (a_1, a_2, \ldots, a_N)$ needs to be chosen satisfying

$$||aL|| = ||a_1l_1 + a_2l_2 + \ldots + a_Nl_N||$$

where

$$L = \begin{bmatrix}
    l_1 \\
    l_2 \\
    \vdots \\
    l_N
\end{bmatrix}$$

• This problem is equivalent to finding the shortest vector in a lattice.

• An approximate solution can be found by LLL algorithm.
Simulation Results
Simulation Procedure

1. Set up channel parameter $H$
2. Find optimal parameters $\alpha$ and $\alpha$
3. Generate messages and encode them
4. Transmit encoded messages
5. Decode received messages and calculate error rate
6. Transmits encoded messages
Simulation Result

![Graph showing simulation results with different cases: Gaussian Integer Case, LBE (Gaussian), Eisenstein Integer Case, LBE (Eisenstein). The graph plots SNR on the x-axis and SEPR (signal error probability) on the y-axis. The lines show how the error probability decreases with increasing SNR.]
Acknowledgement and References

Back matters
Acknowledgement

Thanks to

Prof. Robert LI and Dr. Tyler Sun
Core References

• Qifu Tyler Sun, Jinhong Yuan, Tao Huang, and Kenneth W. Shum, Lattice Network Codes Based on Eisenstein Integers, IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. 61, NO. 7, JULY 2013


Our Progress Compared to Last Term

• Simulation on LNC with at least three users (senders);
• Detailed proof of the performance of LNC.