HERMITE'S INEQUALITY AND THE LLL ALGORITHM

PHONG NGUYEN

http://www.di.ens.fr/~pnguyen

November 2024

OVERVIEW: LATTICE ALGORITHMS

INSIGHT

INSIGHT

- The most classical problem is to prove the existence of short lattice vectors.

INSIGHT

- The most classical problem is to prove the existence of short lattice vectors.
- All known upper bounds on Hermite's constant have an algorithmic analogue:

INSIGHT

- The most classical problem is to prove the existence of short lattice vectors.

- All known upper bounds on Hermite's constant have an algorithmic analogue:
  - Hermite's inequality: **The LLL Algorithm**

INSIGHT

- The most classical problem is to prove the existence of short lattice vectors.
- All known upper bounds on Hermite's constant have an algorithmic analogue:
  - Hermite's inequality: **The LLL Algorithm**
  - Mordell's inequality: **Blockwise generalizations of LLL**

INSIGHT

- The most classical problem is to prove the existence of short lattice vectors.
- All known upper bounds on Hermite's constant have an algorithmic analogue:
  - Hermite's inequality: **The LLL Algorithm**
  - Mordell's inequality: **Blockwise generalizations of LLL**
  - Mordell's proof of Minkowski's inequality:

Worst-case to average-case reductions for SIS and Sieve algorithms [BJN14,ADRS15]

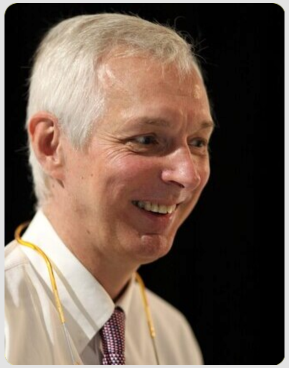
SVP ALGORITHMS

SVP ALGORITHMS

- Poly-time approximation algorithms

SVP ALGORITHMS

- Poly-time approximation algorithms
  - ➤ The LLL algorithm [1982]



Arjen Lenstra

Hendrik Lenstra

László Lovász

SVP ALGORITHMS

- Poly-time approximation algorithms
  - ➤ The LLL algorithm [1982]
  - ➤ Block generalizations by [Schnorr1987], [GHKN06], [GamaN08], [MiWa16]

Claus Peter Schnorr

SVP ALGORITHMS

- Poly-time approximation algorithms **Approximation**
  - The LLL algorithm [1982]
  - Block generalizations by [Schnorr1987], [GHKN06], [GamaN08], [MiWa16]
- Exponential exact algorithms **Exact**

SVP ALGORITHMS

- Poly-time approximation algorithms **Approximation**
  - ➤ The LLL algorithm [1982]
  - ➤ Block generalizations by [Schnorr1987], [GHKN06], [GamaN08], [MiWa16]
- Exponential exact algorithms **Exact**
  - ➤ Poly-space enumeration [Pohst1981,Kannan1983,ScEu1994]

SVP ALGORITHMS

- Poly-time approximation algorithms **Approximation**
  - The LLL algorithm [1982]
  - Block generalizations by [Schnorr1987], [GHKN06], [GamaN08], [MiWa16]
- Exponential exact algorithms **Exact**
  - Poly-space enumeration [Pohst1981,Kannan1983,ScEu1994]
  - Exp-space sieving [AKS01,MV10,ADRS15]

SVP ALGORITHMS

- Poly-time approximation algorithms **Approximation**
  - ➤ The LLL algorithm [1982]
  - ➤ Block generalizations by [Schnorr1987], [GHKN06], [GamaN08], [MiWa16]
- Exponential exact algorithms **Exact**
  - ➤ Poly-space enumeration [Pohst1981,Kannan1983,ScEu1994]
  - ➤ Exp-space sieving [AKS01,MV10,ADRS15]

Both are complementary

TODAY: HERMITE’S INEQUALITY AND THE LLL ALGORITHM

- Proving Hermite's Inequality
- The LLL Algorithm

TODAY: HERMITE'S INEQUALITY AND THE LLL ALGORITHM

- **Proving Hermite's Inequality**

- **The LLL Algorithm**
  - Gram-Schmidt Orthogonalization
  - Size Reduction
  - Hermite Reduction
  - The LLL Algorithm
  - Analysis of LLL

PROVING HERMITE'S INEQUALITY

HERMITE'S INEQUALITY

- Hermite proved in 1850: in any d-rank lattice L, there exists a non-zero v in L s.t.

$$\|\vec{v}\| \leq \left(\frac{4}{3}\right)^{(d-1)/4} \mathrm{vol}(L)^{1/d}$$

HERMITE'S INEQUALITY

- Hermite proved in 1850: in any d-rank lattice L, there exists a non-zero v in L s.t.

$$\|\vec{v}\| \leq \left(\frac{4}{3}\right)^{(d-1)/4} \mathrm{vol}(L)^{1/d}$$

- [LLL82] finds in polynomial time a non-zero lattice vector of norm $\leq (4/3+\varepsilon)^{(d-1)/4}\mathrm{vol}(L)^{1/d}$. It is an algorithmic version of Hermite's inequality.

PROOF OF HERMITE'S INEQUALITY

- Induction over d: obvious for d=1.

PROOF OF HERMITE'S INEQUALITY

- Induction over d: obvious for d=1.

- Let b_1 be a shortest vector of L, and π the projection over $b_1^\perp$.
- Let $\pi(b_2)$ be a shortest vector of $\pi(L)$.

PROOF OF HERMITE'S INEQUALITY

- Induction over d: obvious for d=1.

- Let b_1 be a shortest vector of L, and π the projection over $b_1^\perp$.
- Let $\pi(b_2)$ be a shortest vector of $\pi(L)$.

- We can make sure by lifting that:

  $\|b_2\|^2 \leq \|\pi(b_2)\|^2 + \|b_1\|^2/4$ (size-reduction)

PROOF OF HERMITE'S INEQUALITY

- Induction over d: obvious for d=1.

- Let b_1 be a shortest vector of L, and π the projection over $b_1^\perp$.
- Let $\pi(b_2)$ be a shortest vector of $\pi(L)$.

- We can make sure by lifting that:

  $\|b_2\|^2 \leq \|\pi(b_2)\|^2 + \|b_1\|^2/4$ (size-reduction)

- On the other hand,

  $\|b_1\| \leq \|b_2\|$ and $vol(\pi(L)) = vol(L)/\|b_1\|$.

QUESTION

QUESTION

- Is the proof constructive?

QUESTION

- Is the proof constructive?
- Does it build a non-zero lattice vector satisfying Hermite's inequality:

$$\|\vec{b}_1\| \leq \left(\frac{4}{3}\right)^{(d-1)/4} \mathrm{vol}(L)^{1/d}$$

AN ALGORITHMIC PROOF

- Let b_1 be a primitive vector of L, and π the projection over $b_1^\perp$.
- Find recursively $\pi(b_2) \in \pi(L)$ satisfying Hermite's inequality.

AN ALGORITHMIC PROOF

- Let b_1 be a primitive vector of L, and π the projection over $b_1^{\perp}$.
- Find recursively $\pi(b_2) \in \pi(L)$ satisfying Hermite's inequality.
- Size-reduce so that $||b_2||^2 \leq ||\pi(b_2)||^2 + ||b_1||^2/4$

AN ALGORITHMIC PROOF

- Let b_1 be a primitive vector of L, and π the projection over $b_1^\perp$.
- Find recursively $\pi(b_2) \in \pi(L)$ satisfying Hermite's inequality.
- Size-reduce so that $\|b_2\|^2 \leq \|\pi(b_2)\|^2 + \|b_1\|^2/4$
- If $\|b_2\| < \|b_1\|$, swap(b_1, b_2) and restart, otherwise stop.

AN ALGORITHMIC PROOF

- This algorithm will terminate and output a non-zero lattice vector satisfying Hermite's inequality:

$$\|\vec{b}_1\| \leq \left(\frac{4}{3}\right)^{(d-1)/4} \mathrm{vol}(L)^{1/d}$$

AN ALGORITHMIC PROOF

- This algorithm will terminate and output a non-zero lattice vector satisfying Hermite's inequality:

$$\|\vec{b}_1\| \leq \left(\frac{4}{3}\right)^{(d-1)/4} \mathrm{vol}(L)^{1/d}$$

- But it may not be efficient: LLL does better by strengthening the test $||b_2|| < ||b_1||$.

MORE DETAILS

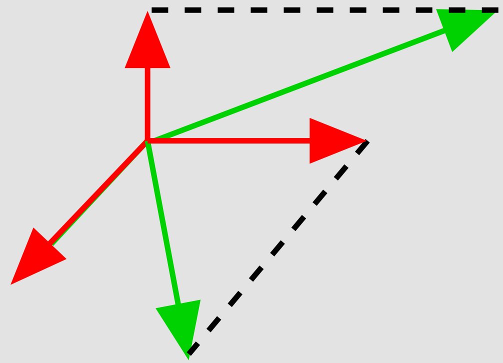
- We need Gram-Schmidt again…

THE LLL ALGORITHM

REMEMBER
GRAM-SCHMIDT ORTHOGONALIZATION

RECALL GRAM-SCHMIDT

- Let $b_1,\ldots,b_n \in \mathbf{R}^m$.
- Its Gram-Schmidt Orthogonalization is $b_1^*,\ldots,b_n^* \in \mathbf{R}^m$ defined as:
    - $b_1^* = b_1$
    - For $2 \le i \le n$, b_i^* = projection of b_i over $\text{span}(b_1,\ldots,b_{i-1})^\perp$



FORMULA

- Let $b_1,\ldots,b_n \in \mathbf{R}^m$ be linearly independent.
- Then all $b_j^* \neq 0$.

FORMULA

- Let $b_1,\ldots,b_n \in \mathbf{R}^m$ be linearly independent.
- Then all $b_j^* \neq 0$.
- For $1 \leq j < i \leq n$, let $$\mu_{i,j} = \frac{\langle \vec{b}_i, \vec{b}_j^* \rangle}{\|\vec{b}_j^*\|^2}$$

FORMULA

- Let $b_1,\ldots,b_n \in \mathbf{R}^m$ be linearly independent.
- Then all $b_j^* \neq 0$.
- For $1 \leq j < i \leq n$, let $$\mu_{i,j} = \frac{\langle \vec{b}_i, \vec{b}_j^* \rangle}{\|\vec{b}_j^*\|^2}$$
- Then: $$\vec{b}_1^\star = \vec{b}_1 \qquad \vec{b}_i^\star = \vec{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \vec{b}_j^\star$$

REMINDER

- If $b_1,\ldots,b_n \in \mathbf{Z}^m$ are linearly independent, we can compute efficiently the rational $\mu_{i,j}$ and $\|\vec{b}_i^*\|^2 \in \mathbb{Q}$, $\vec{b}_i^* \in \mathbb{Q}^m$. More precisely, we can compute the integers $d_i = \mathrm{Gram}(\vec{b}_1, \ldots, \vec{b}_i) = \prod_{j=1}^{i} \|\vec{b}_j^*\|^2$ and $\lambda_{i,j} = d_j \mu_{i,j} \langle \vec{b}_i, \vec{b}_j^* \rangle / \|\vec{b}_j^*\|^2$

$$\vec{b}_i^* = \vec{b}_i - \sum_{j=1}^{i-1} \frac{\lambda_{i,j}}{d_j} \vec{b}_j^*$$

$$\|\vec{b}_i^*\|^2 = \frac{d_i}{d_{i-1}}$$

REMEMBER
SIZE REDUCTION

REMINDER: SIZE-REDUCTION

- Let $b_1,\ldots,b_d \in \mathbf{Z}^m$ be linearly independent.
- $B=(b_1,\ldots,b_d)$ is size-reduced if all $|\mu_{i,j}| \leq \frac{1}{2}$

REMINDER: SIZE-REDUCTION

- Let $b_1,\ldots,b_d \in \mathbf{Z}^m$ be linearly independent.
- $B=(b_1,\ldots,b_d)$ is size-reduced if all $|\mu_{i,j}| \leq \frac{1}{2}$
- Th: There is an efficient algorithm to size-reduce B, without changing the Gram-Schmidt vectors.

VISUALIZING SIZE-REDUCTION

- If we take an appropriate orthonormal basis, the matrix of the lattice basis becomes triangular.

$$\begin{pmatrix} \|\vec{b}_1^*\| & 0 & 0 & \dots & 0 \\ \mu_{2,1}\|\vec{b}_1^*\| & \|\vec{b}_2^*\| & 0 & \dots & 0 \\ \mu_{3,1}\|\vec{b}_1^*\| & \mu_{3,2}\|\vec{b}_2^*\| & \|\vec{b}_3^*\| & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \mu_{d,1}\|\vec{b}_1^*\| & \mu_{d,2}\|\vec{b}_2^*\| & \dots & \mu_{d,d-1}\|\vec{b}_{d-1}^*\| & \|\vec{b}_d^*\| \end{pmatrix}$$

SIZE-REDUCTION ALGORITHM

SIZE-REDUCTION ALGORITHM

- For $i = 2$ to d
  - For $j = i-1$ downto 1
    - ➤ Size-reduce b_i with respect to b_j: make $|\mu_{i,j}| \leq 1/2$ by $b_i := b_i - \text{round}(\mu_{i,j}) b_j$
    - ➤ Update all $\mu_{i,j'}$ for $j' \leq j$.

SIZE-REDUCTION ALGORITHM

- For i = 2 to d
  - For j = i-1 downto 1
    - Size-reduce b_i with respect to b_j: make $|\mu_{i,j}| \leq 1/2$ by $b_i := b_i - \text{round}(\mu_{i,j}) b_j$
    - Update all $\mu_{i,j'}$ for $j' \leq j$.

- The translation does not affect the previous $\mu_{i',j'}$ where $i' < i$, or $i'=i$ and $j'>j$.

HERMITE REDUCTION

HERMITE'S REDUCTION

HERMITE'S REDUCTION

- Hermite proved the existence of bases such that:

$$|\mu_{i,j}| \leq \frac{1}{2} \qquad \text{and} \qquad \frac{\|\vec{b}_i^\star\|^2}{\|\vec{b}_{i+1}^\star\|^2} \leq \frac{4}{3}$$

HERMITE'S REDUCTION

- Hermite proved the existence of bases such that:

$$|\mu_{i,j}| \leq \frac{1}{2} \qquad \text{and} \qquad \frac{\|\vec{b}_i^\star\|^2}{\|\vec{b}_{i+1}^\star\|^2} \leq \frac{4}{3}$$

- Such bases approximate SVP to an exp factor:

$$\|\vec{b}_1\| \leq \left[(4/3)^{1/4}\right]^{d-1} \mathrm{vol}(L)^{1/d} \qquad \gamma_d \leq (4/3)^{(d-1)/2}$$

$$\|\vec{b}_i\| \leq \left[(4/3)^{1/2}\right]^{d-1} \lambda_i(L)$$

GRAPHICALLY

- Condition 1 is over off-diagonal coeffs: size-reduction.
- Condition 2 is over diagonal coeffs.

$$\begin{pmatrix} \|\vec{b}_1^*\| & 0 & 0 & \dots & 0 \\ \mu_{2,1}\|\vec{b}_1^*\| & \|\vec{b}_2^*\| & 0 & \dots & 0 \\ \mu_{3,1}\|\vec{b}_1^*\| & \mu_{3,2}\|\vec{b}_2^*\| & \|\vec{b}_3^*\| & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \mu_{d,1}\|\vec{b}_1^*\| & \mu_{d,2}\|\vec{b}_2^*\| & \dots & \mu_{d,d-1}\|\vec{b}_{d-1}^*\| & \|\vec{b}_d^*\| \end{pmatrix}$$

PROOFS

- Hermite factor: $\|\vec{b}_i^*\| \geq \sqrt{\frac{3}{4}}\|\vec{b}_{i-1}^*\| \geq \ldots \geq \sqrt{\frac{3}{4}}^{i-1} \|\vec{b}_1\|$ so

  $$\text{vol}(L) = \prod_{i=1}^{d} \|\vec{b}_i^*\| \geq \prod_{i=1}^{d} \sqrt{\frac{3}{4}}^{i-1} \|\vec{b}_1\| = \|\vec{b}_1\|^d \sqrt{\frac{3}{4}}^{1+2+\ldots+(d-1)}$$

  therefore $\|\vec{b}_1\| \leq \left(\frac{4}{3}\right)^{(d-1)/4} \text{vol}(L)^{1/d}$

- Approximating SVP:

  $$\lambda_1(L) \geq \min_i \|\vec{b}_i^*\| \geq \min_i \sqrt{\frac{3}{4}}^{i-1} \|\vec{b}_1\| = \sqrt{\frac{3}{4}}^{d-1} \|\vec{b}_1\| \quad \text{so}$$

  $\|\vec{b}_1\| \leq \left(\frac{4}{3}\right)^{(d-1)/2} \lambda_1(L)$. Generalize this to $\|\vec{b}_i\| \leq \left(\frac{4}{3}\right)^{(d-1)/2} \lambda_i(L)$.

COMPUTING HERMITE REDUCTION

COMPUTING HERMITE REDUCTION

- Hermite proved the existence of bases s.t.:

$$|\mu_{i,j}| \leq \frac{1}{2} \quad \text{and} \quad \frac{\|\vec{b}_i^\star\|^2}{\|\vec{b}_{i+1}^\star\|^2} \leq \frac{4}{3}$$

COMPUTING HERMITE REDUCTION

- Hermite proved the existence of bases s.t.:

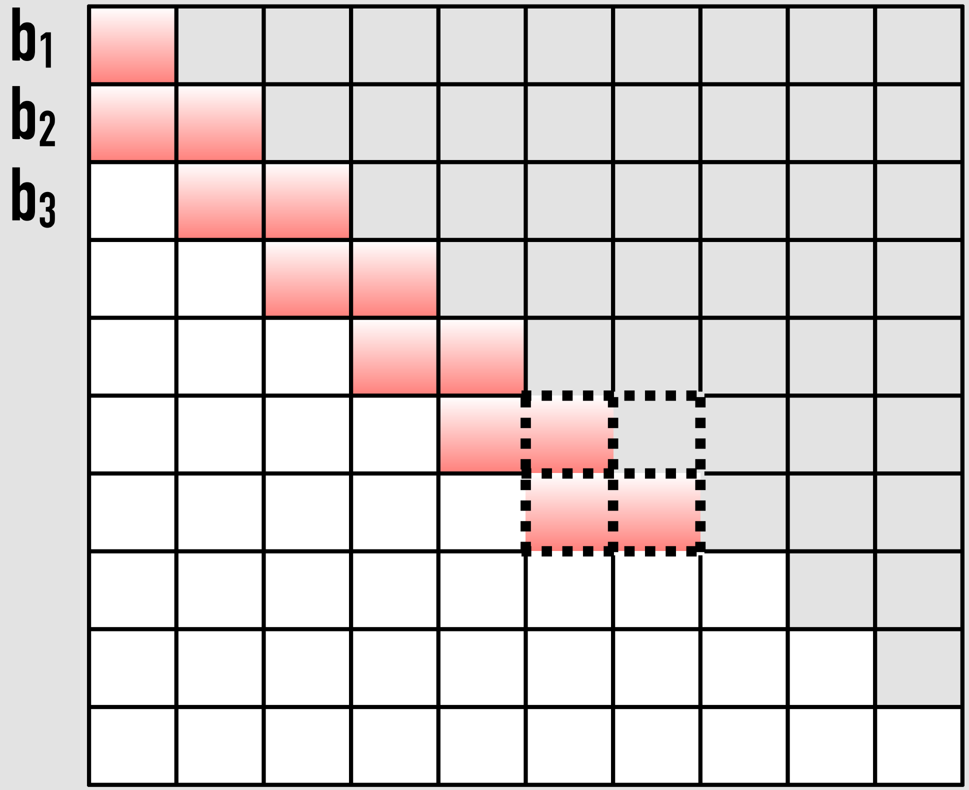
$$|\mu_{i,j}| \leq \frac{1}{2} \quad \text{and} \quad \frac{\|\vec{b}_i^\star\|^2}{\|\vec{b}_{i+1}^\star\|^2} \leq \frac{4}{3}$$

- By relaxing the 4/3, [LLL1982] showed how to compute such a basis in polynomial time.

THE LLL ALGORITHM

HOW LLL WORKS

- LLL is an elegant divide-and-conquer based on 2-dim reduction.



LENSTRA–LENSTRA–LOVÁSZ (LLL) REDUCTION (1982)

LENSTRA–LENSTRA–LOVÁSZ (LLL) REDUCTION (1982)

$$\vec{b}_i^\star = \vec{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \vec{b}_j^\star \qquad \text{where } \mu_{i,j} = \frac{\langle \vec{b}_i, \vec{b}_j^\star \rangle}{\|\vec{b}_j^\star\|^2}$$

LENSTRA-LENSTRA-LOVÁSZ (LLL) REDUCTION (1982)

$$\vec{b}_i^{\star} = \vec{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \vec{b}_j^{\star} \qquad \text{where } \mu_{i,j} = \frac{\langle \vec{b}_i, \vec{b}_j^{\star} \rangle}{\|\vec{b}_j^{\star}\|^2}$$

- A basis is LLL-reduced for $\varepsilon > 0$ if and only if

LENSTRA-LENSTRA-LOVÁSZ (LLL) REDUCTION (1982)

$$\vec{b}_i^\star = \vec{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \vec{b}_j^\star \qquad \text{where } \mu_{i,j} = \frac{\langle \vec{b}_i, \vec{b}_j^\star \rangle}{\|\vec{b}_j^\star\|^2}$$

- A basis is LLL-reduced for $\varepsilon > 0$ if and only if
  - ➤ it is size-reduced $\quad |\mu_{i,j}| \leq \frac{1}{2}$

LENSTRA-LENSTRA-LOVÁSZ (LLL) REDUCTION (1982)

$$\vec{b}_i^\star = \vec{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \vec{b}_j^\star \qquad \text{where } \mu_{i,j} = \frac{\langle \vec{b}_i, \vec{b}_j^\star \rangle}{\|\vec{b}_j^\star\|^2}$$

- A basis is LLL-reduced for $\varepsilon > 0$ if and only if
  - it is size-reduced $\quad |\mu_{i,j}| \leq \frac{1}{2}$
  - Lovász' conditions are satisfied

$$(1-\varepsilon)\|\vec{b}_{i-1}^\star\|^2 \leq \|\vec{b}_i^\star + \mu_{i,i-1}\vec{b}_{i-1}^\star\|^2$$

$$\Rightarrow \|\vec{b}_{i-1}^\star\|^2 \leq \left(\frac{4}{3} + \varepsilon'\right) \|\vec{b}_i^\star\|^2$$

DESCRIPTION OF THE LLL ALGORITHM

- While the basis is not LLL-reduced
  - Size-reduce the basis
  - If Lovász′ condition does not hold for some pair (i-1,i): swap b_{i-1} and b_i.

DESCRIPTION OF THE LLL ALGORITHM

DESCRIPTION OF THE LLL ALGORITHM

- Input: $(b_1,b_2,\ldots,b_d)$ basis of L and $\varepsilon>0$.

DESCRIPTION OF THE LLL ALGORITHM

- Input: $(b_1, b_2, \ldots, b_d)$ basis of L and $\varepsilon > 0$.
- LLL-reduce $(\pi(b_2), \ldots, \pi(b_d))$ where π is the projection over $b_1^\perp$.
- Size-reduce so that $||b_i||^2 \leq ||\pi(b_i)||^2 + ||b_1||^2/4$

DESCRIPTION OF THE LLL ALGORITHM

- Input: $(b_1, b_2, \ldots, b_d)$ basis of L and $\varepsilon > 0$.
- LLL-reduce $(\pi(b_2), \ldots, \pi(b_d))$ where π is the projection over $b_1^{\perp}$.
- Size-reduce so that $||b_i||^2 \leq ||\pi(b_i)||^2 + ||b_1||^2/4$
- If $||b_2|| \leq (1-\varepsilon)||b_1||$, swap$(b_1, b_2)$ and restart, otherwise stop.

ANALYSIS OF LLL

EVOLUTION OF GRAM-SCHMIDT

EVOLUTION OF GRAM-SCHMIDT

- During LLL reduction:

EVOLUTION OF GRAM-SCHMIDT

- During LLL reduction:
  - $\text{Min}_i \, ||b^*_i||$ never decreases.

EVOLUTION OF GRAM-SCHMIDT

- During LLL reduction:
  - $\text{Min}_i \, ||b^*_i||$ never decreases.
  - $\text{Max}_i \, ||b^*_i||$ never increases.

EVOLUTION OF GRAM-SCHMIDT

- During LLL reduction:
  - $\text{Min}_i \, ||b^*_i||$ never decreases.
  - $\text{Max}_i \, ||b^*_i||$ never increases.
  - Each $\text{vol}(b_1,\ldots,b_i)$ never increases.

EVOLUTION OF GRAM-SCHMIDT

- During LLL reduction:
  - $\text{Min}_i \, ||b^*_i||$ never decreases.
  - $\text{Max}_i \, ||b^*_i||$ never increases.
  - Each $\text{vol}(b_1,\ldots,b_i)$ never increases.
  - The only LLL operations that modify the b^*_i's are swaps.

THE IMPACT OF SWAP

THE IMPACT OF SWAP

- We swap b_{i-1} and b_i whenever $(1-\varepsilon)\|\vec{b}^*_{i-1}\|^2 \geq \|\vec{b}^*_i + \mu_{i,i-1}\vec{b}^*_{i-1}\|^2$ which implies that $\|\vec{b}^*_{i-1}\| \geq \|\vec{b}^*_i\|$

THE IMPACT OF SWAP

- We swap b_{i-1} and b_i whenever $(1-\varepsilon)\|\vec{b}^*_{i-1}\|^2 \geq \|\vec{b}^*_i + \mu_{i,i-1}\vec{b}^*_{i-1}\|^2$ which implies that $\|\vec{b}^*_{i-1}\| \geq \|\vec{b}^*_i\|$
- What happens to $\vec{b}^*_{i-1}$ and $\vec{b}^*_i$?

THE IMPACT OF SWAP

- We swap b_{i-1} and b_i whenever $(1-\varepsilon)\|\vec{b}_{i-1}^*\|^2 \geq \|\vec{b}_i^* + \mu_{i,i-1}\vec{b}_{i-1}^*\|^2$ which implies that $\|\vec{b}_{i-1}^*\| \geq \|\vec{b}_i^*\|$
- What happens to $\vec{b}_{i-1}^*$ and $\vec{b}_i^*$?
  - New($\vec{b}_{i-1}^*$)=$\vec{b}_i^* + \mu_{i,i-1}\vec{b}_{i-1}^*$ has norm between $\|\vec{b}_i^*\|$ and $\sqrt{1-\varepsilon}\|\vec{b}_{i-1}^*\|$, hence ≥√(1-ε) shorter.

THE IMPACT OF SWAP

- We swap b_{i-1} and b_i whenever $(1-\varepsilon)\|\vec{b}^*_{i-1}\|^2 \geq \|\vec{b}^*_i + \mu_{i,i-1}\vec{b}^*_{i-1}\|^2$ which implies that $\|\vec{b}^*_{i-1}\| \geq \|\vec{b}^*_i\|$
- What happens to $\vec{b}^*_{i-1}$ and $\vec{b}^*_i$?
  - New($\vec{b}^*_{i-1}$)=$\vec{b}^*_i + \mu_{i,i-1}\vec{b}^*_{i-1}$ has norm between $\|\vec{b}^*_i\|$ and $\sqrt{1-\varepsilon}\|\vec{b}^*_{i-1}\|$, hence ≥√(1-ε) shorter.
  - New($\vec{b}^*_i$) has norm between $\|\vec{b}^*_i\|/\sqrt{1-\varepsilon}$ and $\|\vec{b}^*_{i-1}\|$, hence ≥1/√(1-ε) longer.

THE IMPACT OF SWAP

- We swap b_{i-1} and b_i whenever $(1-\varepsilon)\|\vec{b}^*_{i-1}\|^2 \geq \|\vec{b}^*_i + \mu_{i,i-1}\vec{b}^*_{i-1}\|^2$ which implies that $\|\vec{b}^*_{i-1}\| \geq \|\vec{b}^*_i\|$
- What happens to $\vec{b}^*_{i-1}$ and $\vec{b}^*_i$?
  - New($\vec{b}^*_{i-1}$)=$\vec{b}^*_i + \mu_{i,i-1}\vec{b}^*_{i-1}$ has norm between $\|\vec{b}^*_i\|$ and $\sqrt{1-\varepsilon}\|\vec{b}^*_{i-1}\|$, hence ≥√(1-ε) shorter.
  - New($\vec{b}^*_i$) has norm between $\|\vec{b}^*_i\|/\sqrt{1-\varepsilon}$ and $\|\vec{b}^*_{i-1}\|$, hence ≥1 / √(1-ε) longer.
  - [new($\|\vec{b}^*_i\|$),new($\|\vec{b}^*_{i-1}\|$)] $\subseteq$ [$\|\vec{b}^*_i\|$,$\|\vec{b}^*_{i-1}\|$]

WHY LLL IS POLYNOMIAL

WHY LLL IS POLYNOMIAL

- Consider the quantity

$$P = \prod_{i=1}^{d} \|\vec{b}_i^*\|^{2(d-i+1)}$$

WHY LLL IS POLYNOMIAL

- Consider the quantity

$$P = \prod_{i=1}^{d} \|\vec{b}_i^*\|^{2(d-i+1)}$$

- If the b_i's have integral coordinates, then P is a positive integer.

WHY LLL IS POLYNOMIAL

- Consider the quantity $$P = \prod_{i=1}^{d} \|\vec{b}_i^*\|^{2(d-i+1)}$$

- If the b_i's have integral coordinates, then P is a positive integer.
  - Size-reduction does not modify P.

WHY LLL IS POLYNOMIAL

- Consider the quantity $$P = \prod_{i=1}^{d} \|\vec{b}_i^*\|^{2(d-i+1)}$$
- If the b_i's have integral coordinates, then P is a positive integer.
  - ➤ Size-reduction does not modify P.
  - ➤ But each swap of LLL makes P decrease by a factor <= 1-ε

WHY LLL IS POLYNOMIAL

- Consider the quantity $$P = \prod_{i=1}^{d} \|\vec{b}_i^*\|^{2(d-i+1)}$$
- If the b_i's have integral coordinates, then P is a positive integer.
  - ➤ Size-reduction does not modify P.
  - ➤ But each swap of LLL makes P decrease by a factor <= 1-ε
- This implies that the number of swaps is polynomially bounded.

REMARKS

REMARKS

- We described a simple version of LLL, which is not optimized for implementation.

REMARKS

- We described a simple version of LLL, which is not optimized for implementation.
- We did not fully prove that LLL is polynomial time, because we did not pay attention to the size of all temporary variables.

RECAP OF LLL

RECAP OF LLL

- The LLL algorithm finds in polynomial time a basis such that:

$$|\mu_{i,j}| \leq \frac{1}{2} \qquad \text{and} \qquad \frac{\|\vec{b}_i^\star\|^2}{\|\vec{b}_{i+1}^\star\|^2} \leq \frac{4}{3} + \varepsilon$$

RECAP OF LLL

- The LLL algorithm finds in polynomial time a basis such that:

$$|\mu_{i,j}| \leq \frac{1}{2} \qquad \text{and} \qquad \frac{\|\vec{b}_i^\star\|^2}{\|\vec{b}_{i+1}^\star\|^2} \leq \frac{4}{3} + \varepsilon$$

- Such bases approximate SVP to an exp factor:

$$\|\vec{b}_1\| \leq \left[(4/3+\varepsilon)^{1/4}\right]^{d-1} \mathrm{vol}(L)^{1/d} \qquad \gamma_d \leq (4/3)^{(d-1)/2}$$

$$\|\vec{b}_i\| \leq \left[(4/3+\varepsilon)^{1/2}\right]^{d-1} \lambda_i(L)$$

TAKE AWAY

TAKE AWAY

- Hermite's inequality and LLL are based on two key ideas:

TAKE AWAY

- Hermite's inequality and LLL are based on two key ideas:
  - Projection

TAKE AWAY

- Hermite's inequality and LLL are based on two key ideas:
  - Projection
  - Lifting projected vectors aka size-reduction.

THE MAGIC OF LLL

THE MAGIC OF LLL

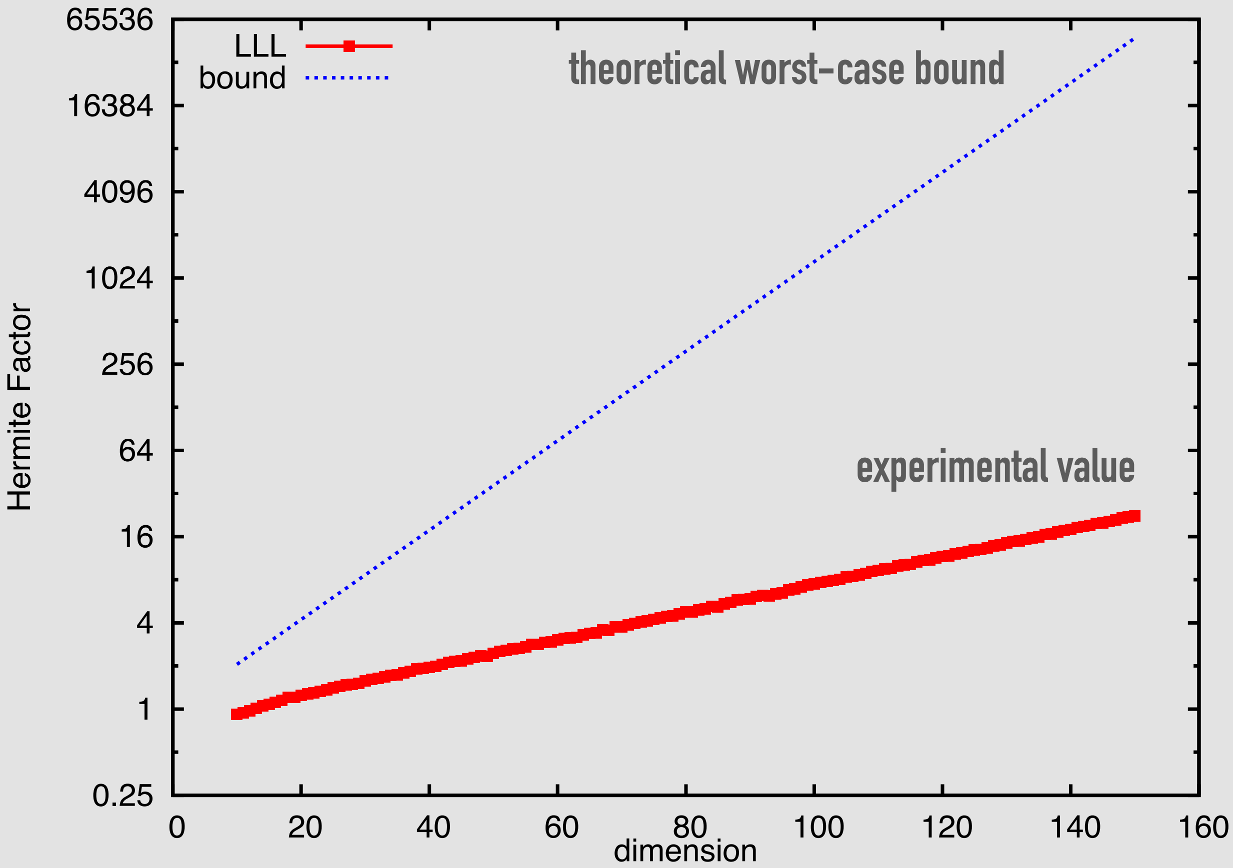
- One of the main reasons behind the popularity of LLL is that it performs "much better" than what the worst-case bounds suggest, especially in low dimension.

THE MAGIC OF LLL

- One of the main reasons behind the popularity of LLL is that it performs “much better” than what the worst-case bounds suggest, especially in low dimension.

- This is another example of worst-case vs. “average-case” and the difficulty of security estimates.

ILLUSTRATION



$$\log \frac{||\vec{b}_1||}{\mathrm{vol}(L)^{1/d}}$$

LLL: THEORY VS PRACTICE

LLL: THEORY VS PRACTICE

- The approx factors $(4/3+\varepsilon)^{(d-1)/4}$ is tight in the worst case and for most LLL bases of a random lattice [KiVe16].

LLL: THEORY VS PRACTICE

- The approx factors $(4/3+\varepsilon)^{(d-1)/4}$ is tight in the worst case and for most LLL bases of a random lattice [KiVe16].

- Experimentally, $4/3+\varepsilon \approx 1.33$ can be replaced by a smaller constant ≈ 1.08, for any lattice, by randomizing the input basis. This means that LLL biases the output distribution.

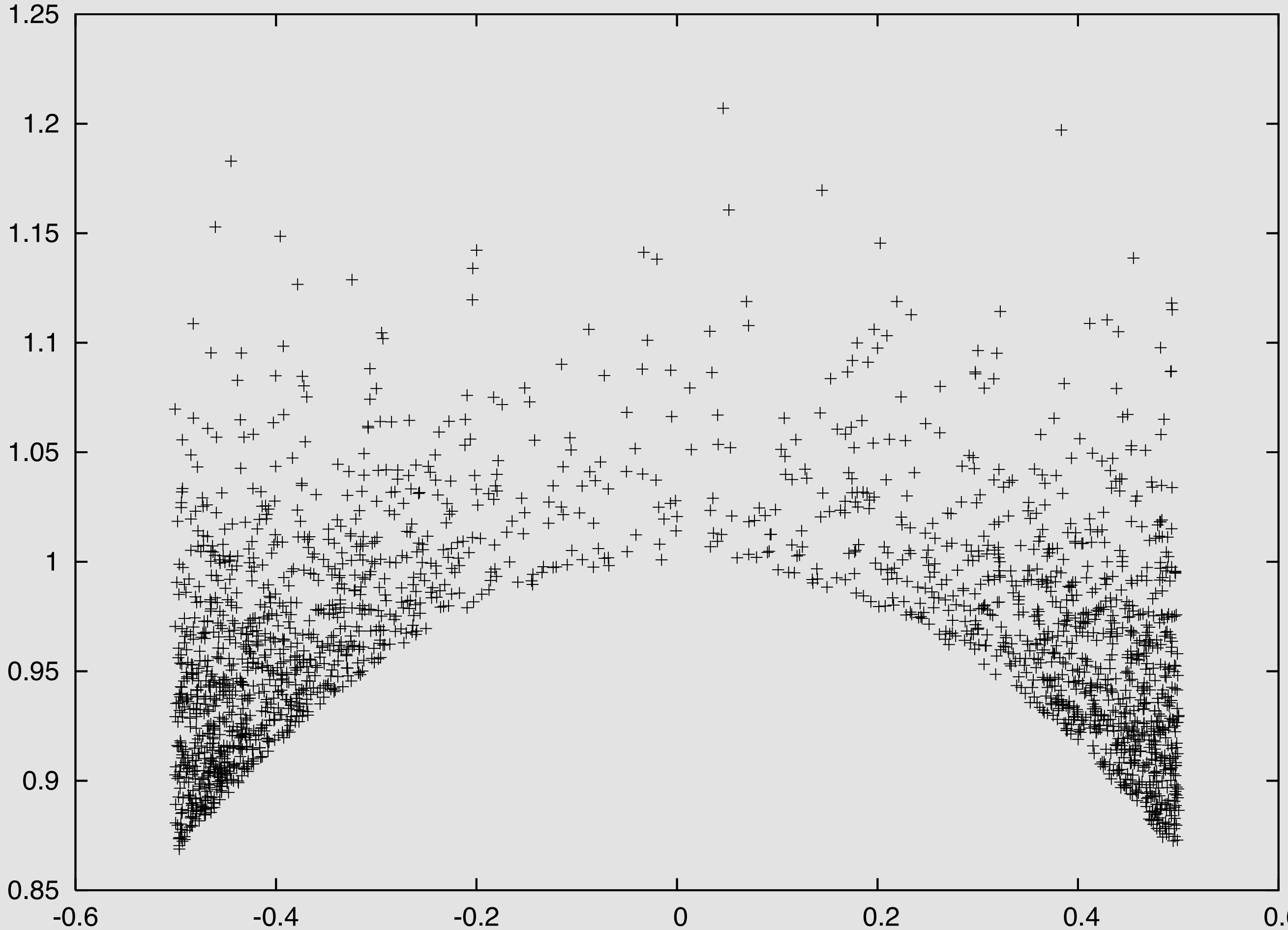
LLL: THEORY VS PRACTICE

- The approx factors $(4/3+\varepsilon)^{(d-1)/4}$ is tight in the worst case and for most LLL bases of a random lattice [KiVe16].
- Experimentally, $4/3+\varepsilon \approx 1.33$ can be replaced by a smaller constant ≈ 1.08, for any lattice, by randomizing the input basis. This means that LLL biases the output distribution.
- No proof for this phenomenon, but…

LLL IN THE REAL WORLD

Distribution of $\vec{b}_i^* + \mu_{i,i-1}\vec{b}_{i-1}^*$

$(\vec{b}_{i-1}^*, \vec{b}_i^* + \mu_{i,i-1}\vec{b}_{i-1}^*)$ *far from densest lattice*



OPEN PROBLEMS

OPEN PROBLEMS

- Take a random integer lattice L.
- Let B be the Hermite normal form of L.

OPEN PROBLEMS

- Take a random integer lattice L.
- Let B be the Hermite normal form of L.

➤ Is is true that with overwhelming probability, after LLL-reducing B, $||b_1|| \leq c^{d-1} vol(L)^{1/d}$ for some $c<(4/3)^{1/4}$?

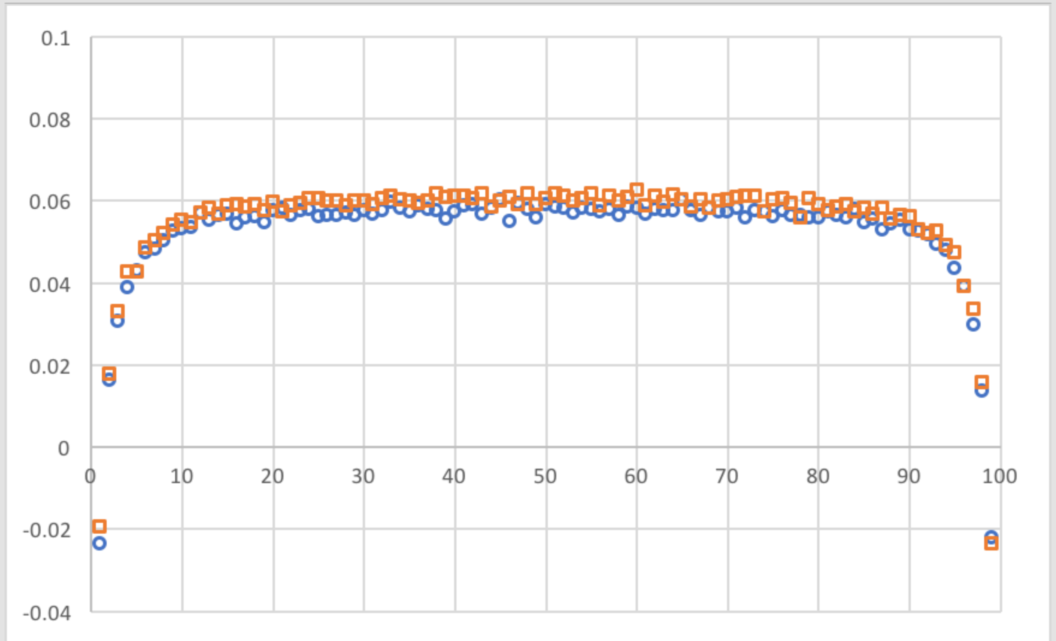
OPEN PROBLEMS

- Take a random integer lattice L.
- Let B be the Hermite normal form of L.

➤ Is is true that with overwhelming probability, after LLL-reducing B, $||b_1|| \leq c^{d-1} vol(L)^{1/d}$ for some $c<(4/3)^{1/4}$?

➤ Can we guess the distribution of $||b_1||$ and the running time?

MODELLING LLL WITH SANDPILES [DKTWY22]



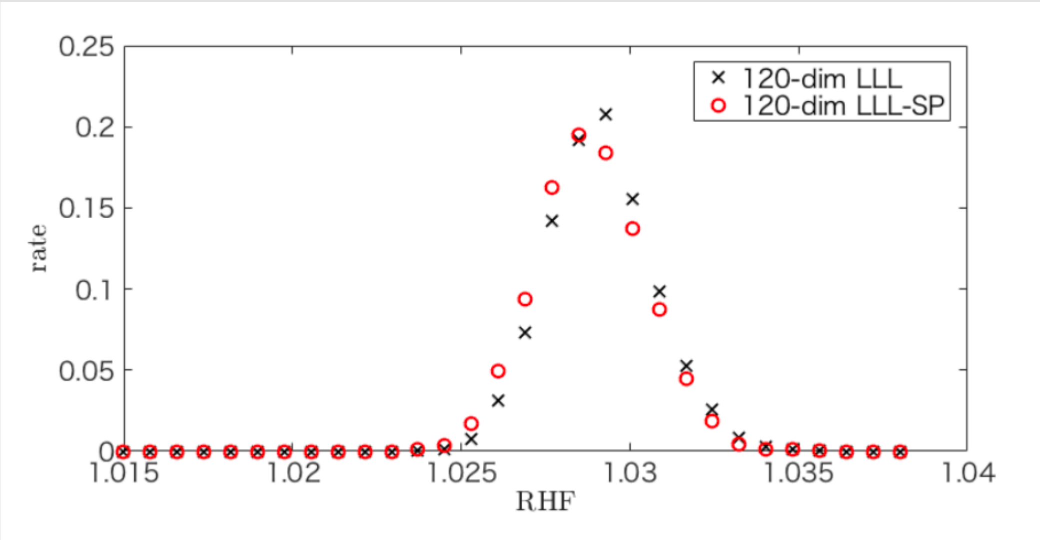
$$\log \frac{\|\vec{b}_i^*\|}{\|\vec{b}_{i+1}^*\|}$$

If there is a swap,

three consecutive values change:

the middle one decreases,

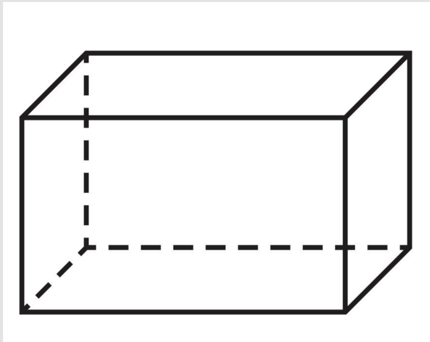
and the other two increase.



$$\left(\frac{\|\vec{b}_1\|}{\mathrm{vol}(L)^{1/d}} \right)^{1/d}$$

BABAI'S NEAREST PLANE ALGORITHM

- Input: a basis $(\vec{b}_1, \ldots, \vec{b}_n)$ of a lattice L and a target $\vec{t}$ in span(L).
- Output: a lattice point $\vec{u}$ such that $\vec{t} - \vec{u} \in \left\{ \sum_{i=1}^{n} x_i \vec{b}_i^\star, -1/2 \leq x_i < 1/2 \right\}$ where the $\vec{b}_i^\star$'s are the Gram-Schmidt orthogonalization.



- For i=n downto 1
  - Compute $\mu_i = \dfrac{\langle \vec{t}, \overrightarrow{b_i}^\star \rangle}{\| \overrightarrow{b_i}^\star \|^2}$
  - $\vec{t} \leftarrow \vec{t} - \lfloor \mu_i \rceil \vec{b}_i$
- Return

$$\vec{u} = \sum_{i=1}^{n} \lfloor \mu_i \rceil \vec{b}_i$$

PAIRING LLL WITH BABAI´S NEAREST PLANE

PAIRING LLL WITH BABAI'S NEAREST PLANE

- Using an LLL-reduced basis, Babai's nearest plane algorithm approximates CVP to within an exponential factor $2\left(\frac{2}{\sqrt{3}}\right)^d$.

$$\|\vec{t} - \vec{u}\|^2 \leq \frac{1}{4}\sum_{i=1}^{d} \|\vec{b}_i^*\|^2 \leq \frac{\|\vec{b}_d^*\|^2}{4}\sum_{i=1}^{d} (\frac{4}{3} + \varepsilon)^{d-i}$$

PAIRING LLL WITH BABAI'S NEAREST PLANE

- Using an LLL-reduced basis, Babai's nearest plane algorithm approximates CVP to within an exponential factor $2\left(\frac{2}{\sqrt{3}}\right)^d$.

$$\|\vec{t}-\vec{u}\|^2 \leq \frac{1}{4}\sum_{i=1}^{d}\|\vec{b}_i^*\|^2 \leq \frac{\|\vec{b}_d^*\|^2}{4}\sum_{i=1}^{d}(\frac{4}{3}+\varepsilon)^{d-i}$$

- Thus, LLL approximates in poly-time both SVP and CVP to within an exponential factor.

BEYOND LLL: SHORTER VECTORS

BEYOND LLL: SHORTER VECTORS

- One can improve upon the LLL approximation factor

$$\left(\frac{4}{3}+\varepsilon\right)^{(d-1)/4} \mathrm{vol}(L)^{1/d} \text{ or } \left(\frac{4}{3}\right)^{(d-1)/2} \lambda_1(L) \text{ in poly-time.}$$

BEYOND LLL: SHORTER VECTORS

- One can improve upon the LLL approximation factor

$$\left(\frac{4}{3}+\varepsilon\right)^{(d-1)/4} \mathrm{vol}(L)^{1/d} \text{ or } \left(\frac{4}{3}\right)^{(d-1)/2} \lambda_1(L) \text{ in poly-time.}$$

- Blockwise algorithms achieve $2^{O\left(d \frac{\log \log d}{\log d}\right)}$

BEYOND LLL: FASTER

- very fast heuristic variants are now available [RyHe23].

BEYOND LLL: FASTER

- LLL is polynomial in the bit-length of the entries.

    - very fast heuristic variants are now available [RyHe23].

BEYOND LLL: FASTER

- LLL is polynomial in the bit-length of the entries.

- It is possible to:

  - very fast heuristic variants are now available [RyHe23].

BEYOND LLL: FASTER

- LLL is polynomial in the bit-length of the entries.

- It is possible to:
  - ➤ make LLL quasi-linear w.r.t. the basis coefficients.
  - ➤ very fast heuristic variants are now available [RyHe23].

BEYOND LLL: FASTER

- LLL is polynomial in the bit-length of the entries.

- It is possible to:
  - make LLL quasi-linear w.r.t. the basis coefficients.
  - decrease the exponents of the time complexity.
  - very fast heuristic variants are now available [RyHe23].

BEYOND LLL: BEYOND Z

BEYOND LLL: BEYOND Z

- LLL is defined for **Z**-bases.

BEYOND LLL: BEYOND Z

- LLL is defined for **Z**-bases.
- Generalizations of LLL are known if:

BEYOND LLL: BEYOND Z

- LLL is defined for **Z**-bases.

- Generalizations of LLL are known if:
  - **Z** is replaced by **Z**[i] or certain rings of integers: this is related to the security of NIST standards.

BEYOND LLL: BEYOND Z

- LLL is defined for **Z**-bases.
- Generalizations of LLL are known if:
  - **Z** is replaced by **Z**[i] or certain rings of integers: this is related to the security of NIST standards.
  - The lattice has additional structure, e.g. symplectic lattices.

BEYOND LLL: BEYOND Z

- LLL is defined for **Z**-bases.
- Generalizations of LLL are known if:
  - **Z** is replaced by **Z**[i] or certain rings of integers: this is related to the security of NIST standards.
  - The lattice has additional structure, e.g. symplectic lattices.
  - But what does it change?