LLL-reducing in quasi-linear time

Damien Stehlé
Joint work with A. Novocin & G. Villard

LIP – CNRS/ENSL/INRIA/UCBL/U. Lyon

Rocquencourt, April 2011

Euclidean lattices

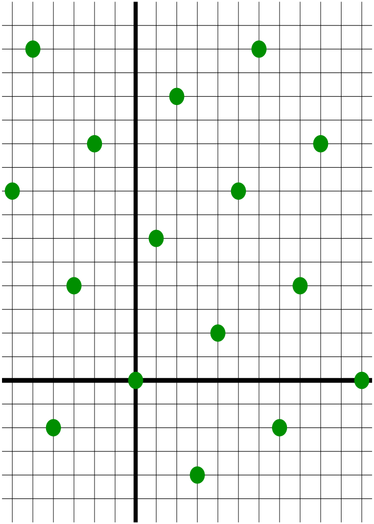
Lattice $\equiv$ discrete subgroup of $\mathbb{R}^n$

$\equiv \quad \{\sum_{i \leq n} x_i \mathbf{b}_i : x_i \in \mathbb{Z}\}$

If the $\mathbf{b}_i$'s are linearly independent, they are called a basis.

Bases are not unique, but they can be obtained from each other by integer transforms of determinant ± 1:

$$\begin{bmatrix} -2 & 1 \\ 10 & 6 \end{bmatrix} = \begin{bmatrix} 4 & -3 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}.$$



Euclidean lattices

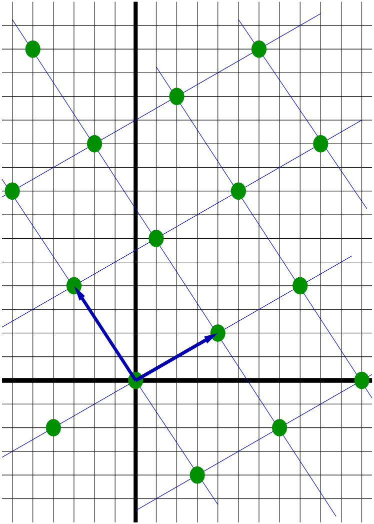
Lattice $\equiv$ discrete subgroup of $\mathbb{R}^n$

$$\equiv \quad \{\sum_{i \leq n} x_i \mathbf{b}_i : x_i \in \mathbb{Z}\}$$

If the $\mathbf{b}_i$'s are linearly independent, they are called a basis.

Bases are not unique, but they can be obtained from each other by integer transforms of determinant ± 1:

$$\begin{bmatrix} -2 & 1 \\ 10 & 6 \end{bmatrix} = \begin{bmatrix} 4 & -3 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}.$$



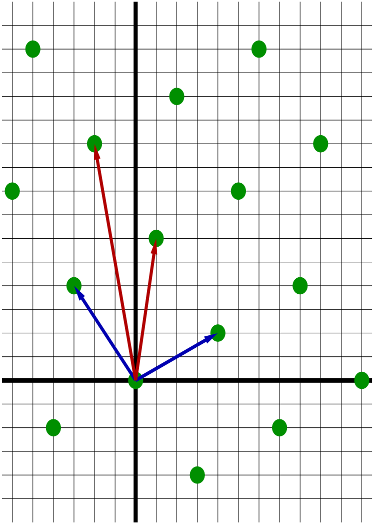
Euclidean lattices

Lattice $\equiv$ discrete subgroup of $\mathbb{R}^n$
$\equiv \{\sum_{i \leq n} x_i \mathbf{b}_i : x_i \in \mathbb{Z}\}$

If the $\mathbf{b}_i$'s are linearly independent, they are called a basis.

Bases are not unique, but they can be obtained from each other by integer transforms of determinant ± 1:

$$\begin{bmatrix} -2 & 1 \\ 10 & 6 \end{bmatrix} = \begin{bmatrix} 4 & -3 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}.$$



Lattice reduction: a representation paradigm

Lattice reduction: Start from an arbitrary basis, and improve the norms/orthogonality of its vectors.

What for?

- Shorter vectors $\Rightarrow$ less space.
- Reduced bases provide intrinsic information about the lattice.
- Reduced bases are easier to compute with.

Lattice reduction as a matrix problem:

Given $B \in \mathbb{R}^{n \times n}$ full-rank, find $U \in GL_n(\mathbb{Z})$ s.t.

BU small and/or with a "nice" QR-factor R.

Lattice reduction: a representation paradigm

Lattice reduction: Start from an arbitrary basis, and improve the norms/orthogonality of its vectors.

What for?

- Shorter vectors $\Rightarrow$ less space.
- Reduced bases provide intrinsic information about the lattice.
- Reduced bases are easier to compute with.

Lattice reduction as a matrix problem:

Given $B \in \mathbb{R}^{n \times n}$ full-rank, find $U \in GL_n(\mathbb{Z})$ s.t.

BU small and/or with a "nice" QR-factor R.

Lattice reduction: a representation paradigm

Lattice reduction: Start from an arbitrary basis, and improve the norms/orthogonality of its vectors.

What for?

- Shorter vectors $\Rightarrow$ less space.
- Reduced bases provide intrinsic information about the lattice.
- Reduced bases are easier to compute with.

Lattice reduction as a matrix problem:

Given $B \in \mathbb{R}^{n \times n}$ full-rank, find $U \in GL_n(\mathbb{Z})$ s.t.

BU small and/or with a "nice" QR-factor R.

Why do we care about lattices?

- Computer algebra: factorisation of rational polynomials.
- Cryptography: cryptanalyses of variants of RSA.
- Communications theory: MIMO, GPS, error correcting codes.
- Combinatorial optimisation, algorithmic group theory, algorithmic number theory, computer arithmetic, etc.

Why do we care about lattices?

- Computer algebra: factorisation of rational polynomials.
- Cryptography: cryptanalyses of variants of RSA.
- Communications theory: MIMO, GPS, error correcting codes.
- Combinatorial optimisation, algorithmic group theory, algorithmic number theory, computer arithmetic, etc.

Lattices tend to pop out every time one wants to use linear algebra but is restricted to discrete transformations.

The LLL reduction [Lenstra-Lenstra-Lovász'82]

Let $\delta \in (1/4, 1)$. A basis $B = (\mathbf{b}_i)_{i \leq n} \in \mathbb{R}^{n \times n}$ with QR-factorisation $B = QR$ is said LLL-reduced if:

- $\forall i, j : \; |r_{i,j}| \leq r_{i,i}/2$ [size-reduction]
- $\forall i : \; \delta \cdot r_{i,i}^2 \leq r_{i,i+1}^2 + r_{i+1,i+1}^2$ [Lovász' condition].

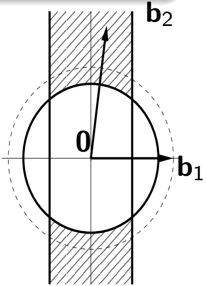
The LLL reduction [Lenstra-Lenstra-Lovász'82]

Let $\delta \in (1/4, 1)$. A basis $B = (\mathbf{b}_i)_{i \leq n} \in \mathbb{R}^{n \times n}$ with QR-factorisation $B = QR$ is said LLL-reduced if:

- $\forall i, j: \ |r_{i,j}| \leq r_{i,i}/2$ [size-reduction]
- $\forall i: \ \delta \cdot r_{i,i}^2 \leq r_{i,i+1}^2 + r_{i+1,i+1}^2$ [Lovász' condition].

The $r_{i,i}$'s can't drop too fast: $r_{i+1,i+1} \geq \sqrt{\delta - \frac{1}{4}}\, r_{i,i}$.
$\prod_i \|\mathbf{b}_i\| \leq 2^{\mathcal{O}(n^2)} \cdot \det(L)$.

$\det(L) := \det(\mathbf{b}_i)_i$ is a lattice invariant.
$\delta < 1$ is crucial to get polynomial-time complexity.



The LLL reduction [Lenstra-Lenstra-Lovász'82]

Let $\delta \in (1/4, 1)$. A basis $B = (\mathbf{b}_i)_{i \leq n} \in \mathbb{R}^{n \times n}$ with QR-factorisation $B = QR$ is said LLL-reduced if:

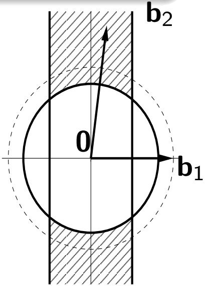
- $\forall i, j : \ |r_{i,j}| \leq r_{i,i}/2$ [size-reduction]
- $\forall i : \ \delta \cdot r_{i,i}^2 \leq r_{i,i+1}^2 + r_{i+1,i+1}^2$ [Lovász' condition].

The $r_{i,i}$'s can't drop too fast: $r_{i+1,i+1} \geq \sqrt{\delta - \frac{1}{4}} r_{i,i}$.

$\prod_i \|\mathbf{b}_i\| \leq 2^{\mathcal{O}(n^2)} \cdot \det(L)$.

$\det(L) := \det(\mathbf{b}_i)_i$ is a lattice invariant.

$\delta < 1$ is crucial to get polynomial-time complexity.



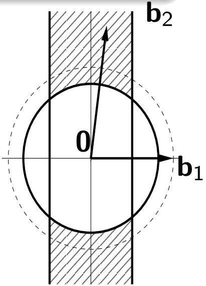
The LLL reduction [Lenstra-Lenstra-Lovász'82]

Let $\delta \in (1/4, 1)$. A basis $B = (\mathbf{b}_i)_{i \leq n} \in \mathbb{R}^{n \times n}$ with QR-factorisation $B = QR$ is said LLL-reduced if:

- $\forall i, j: \ |r_{i,j}| \leq r_{i,i}/2$ [size-reduction]
- $\forall i: \ \delta \cdot r_{i,i}^2 \leq r_{i,i+1}^2 + r_{i+1,i+1}^2$ [Lovász' condition].

The $r_{i,i}$'s can't drop too fast: $r_{i+1,i+1} \geq \sqrt{\delta - \frac{1}{4}} r_{i,i}$.

$\prod_i \|\mathbf{b}_i\| \leq 2^{\mathcal{O}(n^2)} \cdot \det(L)$.



$\det(L) := \det(\mathbf{b}_i)_i$ is a lattice invariant.

$\delta < 1$ is crucial to get polynomial-time complexity.

Complexity bounds

Input: $B \in \mathbb{Z}^{n \times n}$ of full rank, with $\max \|\mathbf{b}_i\| \leq 2^\beta$.

LeLeLo'82	LLL/L^3	$n^{5+\varepsilon}\beta^{2+\varepsilon}$
Kaltofen'83		$n^5\beta^2(n+\beta)^\varepsilon$
Schnorr'87		$n^4\beta(n+\beta)^{1+\varepsilon}$
Nguyen-S.'05	L^2	$n^{4+\varepsilon}\beta(n+\beta)$
Morel-S.-Villard'08	H-LLL	same bound, simpler proof

Can we do better with respect to β?

Complexity bounds

Input: $B \in \mathbb{Z}^{n \times n}$ of full rank, with $\max \|\mathbf{b}_i\| \leq 2^\beta$.

LeLeLo'82	LLL/L^3	$n^{5+\varepsilon}\beta^{2+\varepsilon}$
Kaltofen'83		$n^5\beta^2(n+\beta)^\varepsilon$
Schnorr'87		$n^4\beta(n+\beta)^{1+\varepsilon}$
Nguyen-S.'05	L^2	$n^{4+\varepsilon}\beta(n+\beta)$
Morel-S.-Villard'08	H-LLL	same bound, simpler proof

Can we do better with respect to β?

Quasi-linear LLL-reduction

- Yap'92, Schönhage'91: $\beta^{1+\varepsilon}$ for $n = 2$.
- Eisenbrand-Rote'01: $\beta^{1+\varepsilon}$ for fixed any n.

Our result

We give an algorithm, called $\widetilde{L}^1$, that computes "somewhat" LLL-reduced bases in time $\mathcal{O}(n^{5+\varepsilon}\beta + n^{\omega+1+\varepsilon}\beta^{1+\varepsilon})$.

- n^ω: cost of matrix mult. in dimension n.
- For fixed n: $\mathcal{O}(\mathcal{M}(\beta)\log\beta)$, where $\mathcal{M}(\cdot)$ is for integer mult.
- Same total degree as before.

Quasi-linear LLL-reduction

- Yap'92, Schönhage'91: $\beta^{1+\varepsilon}$ for $n = 2$.
- Eisenbrand-Rote'01: $\beta^{1+\varepsilon}$ for fixed any n.

Our result

We give an algorithm, called $\widetilde{L}^1$, that computes "somewhat" LLL-reduced bases in time $\mathcal{O}(n^{5+\varepsilon}\beta + n^{\omega+1+\varepsilon}\beta^{1+\varepsilon})$.

- n^ω: cost of matrix mult. in dimension n.
- For fixed n: $\mathcal{O}(\mathcal{M}(\beta)\log\beta)$, where $\mathcal{M}(\cdot)$ is for integer mult.
- Same total degree as before.

Plan of the talk

1. **Wishful thinking.**
2. Reducing by deforming.
3. Reducing by truncating.
4. The $\widetilde{L}^1$ algorithm.

A gcd analogy

Euclid's algorithm for computing $\gcd(r_0, r_1)$:

$i := 1$. While $r_i \neq 0$:
Compute $q_i := \lfloor r_{i-1}/r_i \rfloor$, $r_{i+1} := r_{i-1} - q_i r_i$.
Output r_{i-1}.

Vectorial interpretation:

$$\begin{pmatrix} r_i \\ r_{i+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -q_i \end{pmatrix} \cdot \begin{pmatrix} r_{i-1} \\ r_i \end{pmatrix} = \prod_{j=i}^{1} \begin{pmatrix} 0 & 1 \\ 1 & -q_j \end{pmatrix} \cdot \begin{pmatrix} r_0 \\ r_1 \end{pmatrix}$$

LLL as a gcd: Given B_i, find U_i s.t. $B_i U_i$ is closer to reduced.

- L^3: Compute r_{i-1}/r_i exactly before rounding it.
- L^2: Compute r_{i-1}/r_i approximately before rounding it.

A gcd analogy

Euclid's algorithm for computing $\gcd(r_0, r_1)$:

$i := 1$. While $r_i \neq 0$:
  Compute $q_i := \lfloor r_{i-1}/r_i \rfloor$, $r_{i+1} := r_{i-1} - q_i r_i$.
Output r_{i-1}.

Vectorial interpretation:

$$\begin{pmatrix} r_i \\ r_{i+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -q_i \end{pmatrix} \cdot \begin{pmatrix} r_{i-1} \\ r_i \end{pmatrix} = \prod_{j=i}^{1} \begin{pmatrix} 0 & 1 \\ 1 & -q_j \end{pmatrix} \cdot \begin{pmatrix} r_0 \\ r_1 \end{pmatrix}$$

LLL as a gcd: Given B_i, find U_i s.t. $B_i U_i$ is closer to reduced.

- L^3: Compute r_{i-1}/r_i exactly before rounding it.
- L^2: Compute r_{i-1}/r_i approximately before rounding it.

A gcd analogy

Euclid's algorithm for computing $\gcd(r_0, r_1)$:

$i := 1$. While $r_i \neq 0$:
Compute $q_i := \lfloor r_{i-1}/r_i \rfloor$, $r_{i+1} := r_{i-1} - q_i r_i$.
Output r_{i-1}.

Vectorial interpretation:

$$\begin{pmatrix} r_i \\ r_{i+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -q_i \end{pmatrix} \cdot \begin{pmatrix} r_{i-1} \\ r_i \end{pmatrix} = \prod_{j=i}^{1} \begin{pmatrix} 0 & 1 \\ 1 & -q_j \end{pmatrix} \cdot \begin{pmatrix} r_0 \\ r_1 \end{pmatrix}$$

LLL as a gcd: Given B_i, find U_i s.t. $B_i U_i$ is closer to reduced.

- L^3: Compute r_{i-1}/r_i exactly before rounding it.
- L^2: Compute r_{i-1}/r_i approximately before rounding it.

A gcd analogy

Euclid's algorithm for computing $\gcd(r_0, r_1)$:

$i := 1$. While $r_i \neq 0$:
  Compute $q_i := \lfloor r_{i-1}/r_i \rfloor$, $r_{i+1} := r_{i-1} - q_i r_i$.
Output r_{i-1}.

Vectorial interpretation:

$$\begin{pmatrix} r_i \\ r_{i+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -q_i \end{pmatrix} \cdot \begin{pmatrix} r_{i-1} \\ r_i \end{pmatrix} = \prod_{j=i}^{1} \begin{pmatrix} 0 & 1 \\ 1 & -q_j \end{pmatrix} \cdot \begin{pmatrix} r_0 \\ r_1 \end{pmatrix}$$

LLL as a gcd: Given B_i, find U_i s.t. $B_i U_i$ is closer to reduced.

- L^3: Compute r_{i-1}/r_i exactly before rounding it.
- L^2: Compute r_{i-1}/r_i approximately before rounding it.

A gcd analogy

Euclid's algorithm for computing $\gcd(r_0, r_1)$:

$i := 1$. While $r_i \neq 0$:
Compute $q_i := \lfloor r_{i-1}/r_i \rfloor$, $r_{i+1} := r_{i-1} - q_i r_i$.
Output r_{i-1}.

Vectorial interpretation:

$$\begin{pmatrix} r_i \\ r_{i+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -q_i \end{pmatrix} \cdot \begin{pmatrix} r_{i-1} \\ r_i \end{pmatrix} = \prod_{j=i}^{1} \begin{pmatrix} 0 & 1 \\ 1 & -q_j \end{pmatrix} \cdot \begin{pmatrix} r_0 \\ r_1 \end{pmatrix}$$

LLL as a gcd: Given B_i, find U_i s.t. $B_i U_i$ is closer to reduced.

- L^3: Compute r_{i-1}/r_i exactly before rounding it.
- L^2: Compute r_{i-1}/r_i approximately before rounding it.

Towards a quasi-linear time gcd algorithm

Euclid computes remainders $(r_i)_i$ and quotients $(q_i)_i$.

- Assume $r_0 \approx r_1 \approx 2^\beta$.
- Writing down all the r_i's costs $\mathcal{O}(\beta^2)$.

Lehmer'38

If $\frac{|r_0 - \bar{r}_0|}{|r_0|}, \frac{|r_1 - \bar{r}_1|}{|r_1|} \leq 2^{-2\ell}$, then $(q_i)_i$ and $(\bar{q}_i)_i$ share their first ℓ bits.

- Do not compute the q_i's using and updating the lengthy r_i's: Use the shorter $\bar{r}_i$'s instead!
- When the relevant bits of the q_i's are known, apply them to (r_0, r_1)... and apply Lehmer again.
- Knuth'70, Schönhage'71: Do this recursively!

Towards a quasi-linear time gcd algorithm

Euclid computes remainders $(r_i)_i$ and quotients $(q_i)_i$.

- Assume $r_0 \approx r_1 \approx 2^\beta$.
- Writing down all the r_i's costs $\mathcal{O}(\beta^2)$.

Lehmer'38

If $\frac{|r_0 - \bar{r}_0|}{|r_0|}, \frac{|r_1 - \bar{r}_1|}{|r_1|} \leq 2^{-2\ell}$, then $(q_i)_i$ and $(\bar{q}_i)_i$ share their first ℓ bits.

- Do not compute the q_i's using and updating the lengthy r_i's: Use the shorter $\bar{r}_i$'s instead!
- When the relevant bits of the q_i's are known, apply them to (r_0, r_1)... and apply Lehmer again.
- Knuth'70, Schönhage'71: Do this recursively!

Towards a quasi-linear time gcd algorithm

Euclid computes remainders $(r_i)_i$ and quotients $(q_i)_i$.

- Assume $r_0 \approx r_1 \approx 2^\beta$.
- Writing down all the r_i's costs $\mathcal{O}(\beta^2)$.

Lehmer'38

If $\frac{|r_0 - \bar{r}_0|}{|r_0|}, \frac{|r_1 - \bar{r}_1|}{|r_1|} \leq 2^{-2\ell}$, then $(q_i)_i$ and $(\bar{q}_i)_i$ share their first ℓ bits.

- Do not compute the q_i's using and updating the lengthy r_i's: Use the shorter $\bar{r}_i$'s instead!
- When the relevant bits of the q_i's are known, apply them to (r_0, r_1)... and apply Lehmer again.
- Knuth'70, Schönhage'71: Do this recursively!

Towards a quasi-linear time gcd algorithm

Euclid computes remainders $(r_i)_i$ and quotients $(q_i)_i$.

- Assume $r_0 \approx r_1 \approx 2^\beta$.
- Writing down all the r_i's costs $\mathcal{O}(\beta^2)$.

Lehmer'38

If $\frac{|r_0 - \bar{r}_0|}{|r_0|}, \frac{|r_1 - \bar{r}_1|}{|r_1|} \leq 2^{-2\ell}$, then $(q_i)_i$ and $(\bar{q}_i)_i$ share their first ℓ bits.

- Do not compute the q_i's using and updating the lengthy r_i's: Use the shorter $\bar{r}_i$'s instead!
- When the relevant bits of the q_i's are known, apply them to (r_0, r_1)... and apply Lehmer again.
- Knuth'70, Schönhage'71: Do this recursively!

The Knuth-Schönhage gcd algorithm

To compute the first ℓ quotient bits of r_0, r_1 of bit-sizes 2ℓ:

1. Take the first ℓ bits of r_0 and r_1.
2. Recursively get the first $\ell/2$ quotient bits.
3. Apply the quotients to r_0, r_1, to get r_0', r_1'.
4. Take the first ℓ bits of r_0' and r_1'.
5. Recursively get the first $\ell/2$ quotient bits.

- Applying the quotients: multiply a $\mathcal{O}(\ell)$-bit 2×2 matrix to a $\mathcal{O}(\ell)$-bit vector.
- Cost: $C_\ell = 2C_{\ell/2} + \mathcal{O}(\mathcal{M}(\ell)) = \mathcal{O}(\mathcal{M}(\ell) \log \ell)$.
- Can be used to compute gcds in time $\mathcal{O}(\mathcal{M}(\ell) \log \ell)$.

The Knuth-Schönhage gcd algorithm

To compute the first ℓ quotient bits of r_0, r_1 of bit-sizes 2ℓ:

1. Take the first ℓ bits of r_0 and r_1.
2. Recursively get the first $\ell/2$ quotient bits.
3. Apply the quotients to r_0, r_1, to get r_0', r_1'.
4. Take the first ℓ bits of r_0' and r_1'.
5. Recursively get the first $\ell/2$ quotient bits.

- Applying the quotients: multiply a $\mathcal{O}(\ell)$-bit 2×2 matrix to a $\mathcal{O}(\ell)$-bit vector.
- Cost: $C_\ell = 2C_{\ell/2} + \mathcal{O}(\mathcal{M}(\ell)) \;=\; \mathcal{O}(\mathcal{M}(\ell)\log\ell)$.
- Can be used to compute gcds in time $\mathcal{O}(\mathcal{M}(\ell)\log\ell)$.

What about doing it for LLL?

To compute the "first" ℓ bits of U reducing B:

1. Take the first ℓ bits of each b_{ij}.
2. Recursively get the first $\ell/2$ bits of U.
3. Apply them to B, to get a shorter B'.
4. Take the first ℓ bits of each b'_{ij}.
5. Recursively get the next $\ell/2$ bits of U.

- What is a "quotient" here?
- How to control the bit-size of a unimodular matrix?
- Can we truncate "remainders", i.e., lattice bases?
- How to handle multidimensionality / unbalanced magnitudes?

What about doing it for LLL?

To compute the "first" ℓ bits of U reducing B:

1. Take the first ℓ bits of each b_{ij}.
2. Recursively get the first $\ell/2$ bits of U.
3. Apply them to B, to get a shorter B'.
4. Take the first ℓ bits of each b'_{ij}.
5. Recursively get the next $\ell/2$ bits of U.

- What is a "quotient" here?
- How to control the bit-size of a unimodular matrix?
- Can we truncate "remainders", i.e., lattice bases?
- How to handle multidimensionality / unbalanced magnitudes?

Plan of the talk

1. Wishful thinking.
2. **Reducing by deforming.**
3. Reducing by truncating.
4. The $\widetilde{\mathrm{L}}^1$ algorithm.

From reduced to reduced

- If B is arbitrary, then a reducing U can be huge (Cramer :-().
- If B is reduced, any U such that BU is reduced is bounded.

Let B be reduced with R-factor R, and U s.t. BU is reduced. Then:

$$\forall i, j : |u_{ij}| \leq 2^{\mathcal{O}(n)} \cdot r_{jj}/r_{ii}.$$

- If B is reduced, the r_{ii}'s can't decrease fast.
- Assuming they don't increase, we get $\max |u_{ij}| \leq 2^{\mathcal{O}(n)}$.

From reduced to reduced

- If B is arbitrary, then a reducing U can be huge (Cramer :-().
- If B is reduced, any U such that BU is reduced is bounded.

Let B be reduced with R-factor R, and U s.t. BU is reduced. Then:

$$\forall i, j : |u_{ij}| \leq 2^{\mathcal{O}(n)} \cdot r_{jj}/r_{ii}.$$

- If B is reduced, the r_{ii}'s can't decrease fast.
- Assuming they don't increase, we get $\max |u_{ij}| \leq 2^{\mathcal{O}(n)}$.

From reduced to reduced

- If B is arbitrary, then a reducing U can be huge (Cramer :-().
- If B is reduced, any U such that BU is reduced is bounded.

Let B be reduced with R-factor R, and U s.t. BU is reduced. Then:

$$\forall i, j : |u_{ij}| \leq 2^{\mathcal{O}(n)} \cdot r_{jj}/r_{ii}.$$

- If B is reduced, the r_{ii}'s can't decrease fast.
- Assuming they don't increase, we get $\max |u_{ij}| \leq 2^{\mathcal{O}(n)}$.

From reduced to deformed to reduced

- Start from something reduced, deform it a bit, and reduce it!
- The Belabas-van Hoeij-Novocin deformation:

$$B \mapsto \mathrm{diag}(2^\ell, 1, \ldots, 1) \cdot B = \sigma_\ell B.$$

- The r_{ii}'s cannot decrease.
- Their product increases by a factor 2^ℓ.

Let $\ell \geq 0$, B be reduced with R-factor R, and U s.t. $\sigma_\ell BU$ is reduced. Then:

$$\forall i, j : |u_{ij}| \leq 2^{\ell + \mathcal{O}(n)} \cdot r_{jj} / r_{ii}.$$

$\longrightarrow$ If B is "balanced", each u_{ij} has at most $\ell + \mathcal{O}(n)$ bits.

From reduced to deformed to reduced

- Start from something reduced, deform it a bit, and reduce it!
- The Belabas-van Hoeij-Novocin deformation:

$$B \mapsto \mathrm{diag}(2^\ell, 1, \ldots, 1) \cdot B = \sigma_\ell B.$$

- The r_{ii}'s cannot decrease.
- Their product increases by a factor 2^ℓ.

Let $\ell \geq 0$, B be reduced with R-factor R, and U s.t. $\sigma_\ell BU$ is reduced. Then:

$$\forall i, j : |u_{ij}| \leq 2^{\ell + \mathcal{O}(n)} \cdot r_{jj}/r_{ii}.$$

$\longrightarrow$ If B is "balanced", each u_{ij} has at most $\ell + \mathcal{O}(n)$ bits.

From reduced to deformed to reduced

- Start from something reduced, deform it a bit, and reduce it!
- The Belabas-van Hoeij-Novocin deformation:

$$B \mapsto \text{diag}(2^\ell, 1, \ldots, 1) \cdot B = \sigma_\ell B.$$

- The r_{ii}'s cannot decrease.
- Their product increases by a factor 2^ℓ.

Let $\ell \geq 0$, B be reduced with R-factor R, and U s.t. $\sigma_\ell BU$ is reduced. Then:

$$\forall i, j : |u_{ij}| \leq 2^{\ell + \mathcal{O}(n)} \cdot r_{jj} / r_{ii}.$$

$\longrightarrow$ If B is "balanced", each u_{ij} has at most $\ell + \mathcal{O}(n)$ bits.

From reduced to deformed to reduced

- Start from something reduced, deform it a bit, and reduce it!
- The Belabas-van Hoeij-Novocin deformation:

$$B \mapsto \operatorname{diag}(2^{\ell}, 1, \ldots, 1) \cdot B = \sigma_{\ell} B.$$

- The r_{ii}'s cannot decrease.
- Their product increases by a factor 2^{ℓ}.

Let $\ell \geq 0$, B be reduced with R-factor R, and U s.t. $\sigma_{\ell} B U$ is reduced. Then:

$$\forall i, j : |u_{ij}| \leq 2^{\ell + \mathcal{O}(n)} \cdot r_{jj} / r_{ii}.$$

$\longrightarrow$ If B is "balanced", each u_{ij} has at most $\ell + \mathcal{O}(n)$ bits.

Lift-reducing suffices for reducing

Assume $B \in \mathbb{Z}^{n \times n}$ is upper triangular.

$$\begin{bmatrix} b_{1,1} & \dots & b_{1,n-2} & b_{1,n-1} & b_{1,n} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & b_{n-2,n-2} & b_{n-2,n-1} & b_{n-2,n} \\ 0 & \dots & 0 & b_{n-1,n-1} & b_{n-1,n} \\ 0 & \dots & 0 & 0 & b_{n,n} \end{bmatrix}$$

Lift-reducing suffices for reducing

Assume $B \in \mathbb{Z}^{n \times n}$ is upper triangular.

$$\left[\begin{array}{cccc|c} b_{1,1} & \dots & b_{1,n-2} & b_{1,n-1} & b_{1,n} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & b_{n-2,n-2} & b_{n-2,n-1} & b_{n-2,n} \\ 0 & \dots & 0 & b_{n-1,n-1} & b_{n-1,n} \\ \hline 0 & \dots & 0 & 0 & b_{n,n} \end{array}\right]$$

Bottom right 1×1 submatrix is reduced.

Lift-reducing suffices for reducing

Assume $B \in \mathbb{Z}^{n \times n}$ is upper triangular.

$$\left[\begin{array}{ccc|cc} b_{1,1} & \dots & b_{1,n-2} & b_{1,n-1} & b_{1,n} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & b_{n-2,n-2} & b_{n-2,n-1} & b_{n-2,n} \\ \hline 0 & \dots & 0 & \frac{b_{n-1,n-1}}{2^\ell} & \frac{b_{n-1,n}}{2^\ell} \\ 0 & \dots & 0 & 0 & b_{n,n} \end{array}\right]$$

Scale down row $n-1$ so that bottom-right 2×2 submatrix is reduced: $\ell \approx \log b_{n-1,n-1}$.

Lift-reducing suffices for reducing

Assume $B \in \mathbb{Z}^{n\times n}$ is upper triangular.

$$\left[\begin{array}{ccc|cc} b_{1,1} & \dots & b_{1,n-2} & b_{1,n-1} & b_{1,n} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & b_{n-2,n-2} & b_{n-2,n-1} & b_{n-2,n} \\ \hline 0 & \dots & 0 & b_{n-1,n-1} & b_{n-1,n} \\ 0 & \dots & 0 & 0 & b_{n,n} \end{array}\right]$$

Lift row $n-1$ by ℓ bits and reduce bottom-right 2×2 submatrix.

Lift-reducing suffices for reducing

Assume $B \in \mathbb{Z}^{n \times n}$ is upper triangular.

$$\left[\begin{array}{ccc|cc} b_{1,1} & \ldots & b_{1,n-2} & b_{1,n-1} & b_{1,n} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \ldots & b_{n-2,n-2} & b_{n-2,n-1} & b_{n-2,n} \\ \hline 0 & \ldots & 0 & x & x \\ 0 & \ldots & 0 & x & x \end{array}\right]$$

Lift row $n-1$ by ℓ bits and reduce bottom-right 2×2 submatrix.

Lift-reducing suffices for reducing

Assume $B \in \mathbb{Z}^{n \times n}$ is upper triangular.

$$\left[\begin{array}{ccc|cc} b_{1,1} & \dots & b_{1,n-2} & x & x \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & b_{n-2,n-2} & x & x \\ 0 & \dots & 0 & x & x \\ 0 & \dots & 0 & x & x \end{array}\right]$$

Propagate the transformations to the first $n-2$ coordinates, and reduce wrt the diagonal coefficients.

Lift-reducing suffices for reducing

Assume $B \in \mathbb{Z}^{n\times n}$ is upper triangular.

$$\left[\begin{array}{cc|ccc} b_{1,1} & \cdots & b_{1,n-2} & x & x \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ \hline 0 & \cdots & \frac{b_{n-2,n-2}}{2^\ell} & \frac{x}{2^\ell} & \frac{x}{2^\ell} \\ 0 & \cdots & 0 & x & x \\ 0 & \cdots & 0 & x & x \end{array}\right]$$

Scale down row $n-2$ so that bottom-right 3×3 submatrix is reduced: $\ell \approx \log b_{n-2,n-2}$.

Lift-reducing suffices for reducing

Assume $B \in \mathbb{Z}^{n \times n}$ is upper triangular.

$$\left[\begin{array}{cc|ccc} b_{1,1} & \dots & b_{1,n-2} & x & x \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ \hline 0 & \dots & b_{n-2,n-2} & x & x \\ 0 & \dots & 0 & x & x \\ 0 & \dots & 0 & x & x \end{array}\right]$$

Lift row $n-2$ by ℓ bits and reduce bottom-right 3×3 submatrix.

Lift-reducing suffices for reducing

Assume $B \in \mathbb{Z}^{n \times n}$ is upper triangular.

$$\left[\begin{array}{cc|ccc} b_{1,1} & \dots & b_{1,n-2} & x & x \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ \hline 0 & \dots & x & x & x \\ 0 & \dots & x & x & x \\ 0 & \dots & x & x & x \end{array}\right]$$

Lift row $n-2$ by ℓ bits and reduce bottom-right 3×3 submatrix.

Lift-reducing suffices for reducing

Assume $B \in \mathbb{Z}^{n \times n}$ is upper triangular.

$$\left[\begin{array}{cc|ccc} b_{1,1} & \dots & x & x & x \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & x & x & x \\ 0 & \dots & x & x & x \\ 0 & \dots & x & x & x \end{array}\right]$$

Propagate the transformations to the first $n-3$ coordinates, and reduce wrt the diagonal coefficients.

Lift-reducing suffices for reducing

Assume $B \in \mathbb{Z}^{n\times n}$ is upper triangular.

$$\left[\begin{array}{cc|ccc} b_{1,1} & \dots & x & x & x \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & x & x & x \\ 0 & \dots & x & x & x \\ 0 & \dots & x & x & x \end{array}\right]$$

Keep going.

Lift-reducing in quasi-linear time suffices

- McCurley-Hafner'91:
  $H = \mathrm{HNF}(B)$ can be computed in time $\mathcal{O}(n^{\omega+1+\varepsilon}\beta^{1+\varepsilon})$.
- Cost of the lifts:
  $$\mathcal{P}oly(n) \cdot \left(\widetilde{O}(\log h_{n,n}) + \widetilde{O}(\log h_{n-1,n-1}) + \ldots\right)$$
  $$= \mathcal{P}oly(n) \cdot \widetilde{O}(\log \det H)$$
  $$= \mathcal{P}oly(n) \cdot \widetilde{O}(\log \det B).$$
  (in fact, we do a bit better than that)
- Cost of the propagations bounded using the smallness of the transforms: $\mathcal{O}(n^{\omega+1+\varepsilon}(\beta^{1+\varepsilon} + n))$.

Lift-reducing in quasi-linear time suffices

- McCurley-Hafner'91:
  $H = \mathrm{HNF}(B)$ can be computed in time $\mathcal{O}(n^{\omega+1+\varepsilon}\beta^{1+\varepsilon})$.
- Cost of the lifts:

$$\begin{aligned}\mathcal{P}oly(n) \cdot \left(\widetilde{O}(\log h_{n,n}) + \widetilde{O}(\log h_{n-1,n-1}) + \ldots\right) \\ = \mathcal{P}oly(n) \cdot \widetilde{O}(\log \det H) \\ = \mathcal{P}oly(n) \cdot \widetilde{O}(\log \det B).\end{aligned}$$

  (in fact, we do a bit better than that)
- Cost of the propagations bounded using the smallness of the transforms: $\mathcal{O}(n^{\omega+1+\varepsilon}(\beta^{1+\varepsilon} + n))$.

Lift-reducing in quasi-linear time suffices

- McCurley-Hafner'91:
  $H = \mathrm{HNF}(B)$ can be computed in time $\mathcal{O}(n^{\omega+1+\varepsilon}\beta^{1+\varepsilon})$.
- Cost of the lifts:

$$\begin{aligned} \mathcal{P}oly(n) \cdot \Big(\widetilde{O}(\log h_{n,n}) + \widetilde{O}(\log h_{n-1,n-1}) + \ldots\Big) \\ = \mathcal{P}oly(n) \cdot \widetilde{O}(\log \det H) \\ = \mathcal{P}oly(n) \cdot \widetilde{O}(\log \det B). \end{aligned}$$

  (in fact, we do a bit better than that)
- Cost of the propagations bounded using the smallness of the transforms: $\mathcal{O}(n^{\omega+1+\varepsilon}(\beta^{1+\varepsilon} + n))$.

Where are we now?

- LLL-reduction $\longrightarrow$ sequence of Lift-reductions.
- We are to lift-reduce in quasi-linear time.
- More precisely: given ℓ and B reduced, we will find U unimodular such that $\sigma_\ell BU$ is reduced, in time $\widetilde{O}(\ell)$.
- This is independent from the bit-size of B.

- The "LLL quotients" are the matrices U that achieve some amount ℓ of lifting.
- The quotients have bounded magnitudes.
- If B is "balanced", then they have small bit-sizes.

Where are we now?

- LLL-reduction $\longrightarrow$ sequence of Lift-reductions.
- We are to lift-reduce in quasi-linear time.
- More precisely: given ℓ and B reduced, we will find U unimodular such that $\sigma_\ell BU$ is reduced, in time $\widetilde{O}(\ell)$.
- This is independent from the bit-size of B.
- The "LLL quotients" are the matrices U that achieve some amount ℓ of lifting.
- The quotients have bounded magnitudes.
- If B is "balanced", then they have small bit-sizes.

Where are we now?

- LLL-reduction $\longrightarrow$ sequence of Lift-reductions.
- We are to lift-reduce in quasi-linear time.
- More precisely: given ℓ and B reduced, we will find U unimodular such that $\sigma_\ell BU$ is reduced, in time $\widetilde{O}(\ell)$.
- This is independent from the bit-size of B.
- The "LLL quotients" are the matrices U that achieve some amount ℓ of lifting.
- The quotients have bounded magnitudes.
- If B is "balanced", then they have small bit-sizes.

Where are we now?

- LLL-reduction $\longrightarrow$ sequence of Lift-reductions.
- We are to lift-reduce in quasi-linear time.
- More precisely: given ℓ and B reduced, we will find U unimodular such that $\sigma_\ell B U$ is reduced, in time $\widetilde{O}(\ell)$.
- This is independent from the bit-size of B.

- The "LLL quotients" are the matrices U that achieve some amount ℓ of lifting.
- The quotients have bounded magnitudes.
- If B is "balanced", then they have small bit-sizes.

Plan of the talk

1. Wishful thinking.
2. Reducing by deforming.
3. **Reducing by truncating.**
4. The $\widetilde{L}^1$ algorithm.

The LLL-reduction is inappropriate for truncations

The LLL-reduction is inappropriate for truncations

We can't decide reducedness by looking at the (53) top-most bits:

The LLL-reduction is inappropriate for truncations

We can't decide reducedness by looking at the (53) top-most bits:

$$\underset{\text{Not reduced}}{\begin{bmatrix} 1 & 2^{60}+2^5 \\ -1 & 2^{60} \end{bmatrix}} \quad \Longrightarrow \quad \underset{\text{Reduced}}{\begin{bmatrix} 1 & 2^{60} \\ -1 & 2^{60} \end{bmatrix}}$$

$$\underset{\text{Reduced}}{\begin{bmatrix} 1 & 2^{53}+2^{-1}+2^{-25} \\ 2^{-10} & -2^{63} \end{bmatrix}} \quad \Longrightarrow \quad \underset{\text{Not reduced}}{\begin{bmatrix} 1 & 2^{53}+1 \\ 2^{-10} & -2^{63} \end{bmatrix}}$$

The LLL-reduction is inappropriate for truncations

We can't decide reducedness by looking at the (53) top-most bits:

$$\begin{bmatrix} 1 & 2^{60}+2^5 \\ -1 & 2^{60} \end{bmatrix} \Longrightarrow \begin{bmatrix} 1 & 2^{60} \\ -1 & 2^{60} \end{bmatrix}$$

Not reduced $\qquad$ Reduced

$$\begin{bmatrix} 1 & 2^{53}+2^{-1}+2^{-25} \\ 2^{-10} & -2^{63} \end{bmatrix} \Longrightarrow \begin{bmatrix} 1 & 2^{53}+1 \\ 2^{-10} & -2^{63} \end{bmatrix}$$

Reduced $\qquad$ Not reduced

The LLL-reduction is inappropriate for truncations

We can't decide reducedness by looking at the (53) top-most bits:

$$\underbrace{\begin{bmatrix} 1 & 2^{60}+2^{5} \\ -1 & 2^{60} \end{bmatrix}}_{\text{Not reduced}} \Longrightarrow \underbrace{\begin{bmatrix} 1 & 2^{60} \\ -1 & 2^{60} \end{bmatrix}}_{\text{Reduced}}$$

$$\underbrace{\begin{bmatrix} 1 & 2^{53}+2^{-1}+2^{-25} \\ 2^{-10} & -2^{63} \end{bmatrix}}_{\text{Reduced}} \Longrightarrow \underbrace{\begin{bmatrix} 1 & 2^{53}+1 \\ 2^{-10} & -2^{63} \end{bmatrix}}_{\text{Not reduced}}$$

If $B \in \mathbb{Z}^{n\times n}$, we may need all the bits to decide.

If $B \in \mathbb{R}^{n\times n}$, we may not even be able to tell!

Sensitivity of the R-factor

- Take $B \in \mathbb{R}^{n \times n}$ full-rank, with $B = QR$.
- Apply a columnwise perturbation ΔB, i.e., $\max_i \frac{\|\Delta \mathbf{b}_i\|}{\|\mathbf{b}_i\|} \leq \varepsilon$.
- If ε is very small, then $B + \Delta B$ is full-rank and:

$$B + \Delta B = (Q + \Delta Q)(R + \Delta R).$$

- How large can ΔR be?

Sensitivity of the R-factor

- Take $B \in \mathbb{R}^{n \times n}$ full-rank, with $B = QR$.
- Apply a columnwise perturbation ΔB, i.e., $\max_i \frac{\|\Delta \mathbf{b}_i\|}{\|\mathbf{b}_i\|} \leq \varepsilon$.
- If ε is very small, then $B + \Delta B$ is full-rank and:

$$B + \Delta B = (Q + \Delta Q)(R + \Delta R).$$

- How large can ΔR be?

Chang-S-Villard'11

Let $\mathrm{cond}(R) = \| |R||R^{-1}| \|$. If $\mathrm{cond}(R) \cdot \varepsilon \lesssim 1$, then:
$B + \Delta B$ is full-rank and $\max \frac{\|\Delta \mathbf{r}_i\|}{\|\mathbf{r}_i\|} \lesssim \mathrm{cond}(R) \cdot \varepsilon$.

Furthermore, if B is LLL-reduced, then $\mathrm{cond}(R) = 2^{\mathcal{O}(n)}$.

Fixing the LLL-reduction

- We would like the reduction to resist perturbations.
- The bound on $\|\Delta \mathbf{r}_j\|$ is proportional to $\|\mathbf{r}_j\|$.
- By reducedness, $1 \leq \frac{\|\mathbf{r}_j\|}{r_{j,j}} \leq 2^{\mathcal{O}(n)}$.

$\Rightarrow$ $r_{i,j}$ should be related to $r_{j,j}$ instead of (only) $r_{i,i}$.

Fixing the LLL-reduction

- We would like the reduction to resist perturbations.
- The bound on $\|\Delta \mathbf{r}_j\|$ is proportional to $\|\mathbf{r}_j\|$.
- By reducedness, $1 \leq \frac{\|\mathbf{r}_j\|}{r_{j,j}} \leq 2^{\mathcal{O}(n)}$.

$\Rightarrow$ $r_{i,j}$ should be related to $r_{j,j}$ instead of (only) $r_{i,i}$.

Fixing the LLL-reduction

- We would like the reduction to resist perturbations.
- The bound on $\|\Delta \mathbf{r}_j\|$ is proportional to $\|\mathbf{r}_j\|$.
- By reducedness, $1 \leq \frac{\|\mathbf{r}_j\|}{r_{j,j}} \leq 2^{\mathcal{O}(n)}$.

$\Rightarrow$ $r_{i,j}$ should be related to $r_{j,j}$ instead of (only) $r_{i,i}$.

Let $\Xi = (\delta, \eta, \theta)$ with $\eta \in (1/2, 1)$, $\theta > 0$ and $\delta \in (\eta^2, 1)$.
A basis $B \in \mathbb{R}^{n \times n}$ with R-factor R is said Ξ-reduced if:

- $\forall i, j : \ |r_{i,j}| \leq \eta \cdot r_{i,i} + \theta \cdot r_{j,j}$ [Modified size-reduction]
- $\forall i : \ \delta \cdot r_{i,i}^2 \leq r_{i,i+1}^2 + r_{i+1,i+1}^2$.

Fixing the LLL-reduction

- We would like the reduction to resist perturbations.
- The bound on $\|\Delta \mathbf{r}_j\|$ is proportional to $\|\mathbf{r}_j\|$.
- By reducedness, $1 \leq \frac{\|\mathbf{r}_j\|}{r_{j,j}} \leq 2^{\mathcal{O}(n)}$.

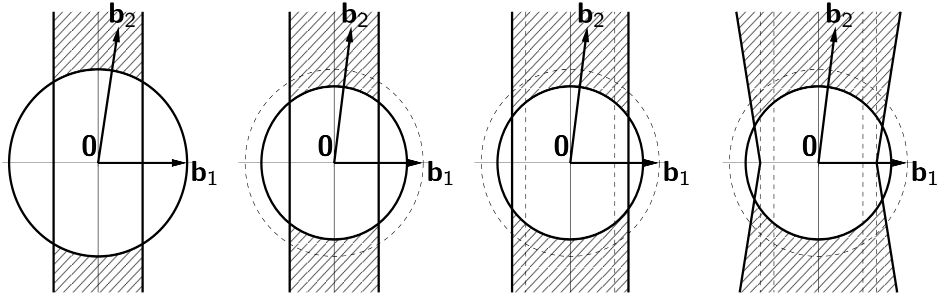
$\Rightarrow$ $r_{i,j}$ should be related to $r_{j,j}$ instead of (only) $r_{i,i}$.

Let $\Xi = (\delta, \eta, \theta)$ with $\eta \in (1/2, 1)$, $\theta > 0$ and $\delta \in (\eta^2, 1)$.
A basis $B \in \mathbb{R}^{n \times n}$ with R-factor R is said Ξ-reduced if:

- $\forall i, j: \ |r_{i,j}| \leq \eta \cdot r_{i,i} + \theta \cdot r_{j,j}$ [Modified size-reduction]
- $\forall i: \ \delta \cdot r_{i,i}^2 \leq r_{i,i+1}^2 + r_{i+1,i+1}^2$.

If B is balanced, this is the same as before.

The LLL-reductions, graphically



$(1, 1/2, 0)$	$(\delta, 1/2, 0)$	$(\delta, \eta, 0)$	(δ, η, θ)
Hermite	LLL'82	Schnorr'88	Chang-S-Villard'11

Properties of the new reduction

The new reduction is perturbation-friendly:

- We still have $\mathrm{cond}(R) = 2^{\mathcal{O}(n)}$ for Ξ-reduced bases.
- If B is reduced and $\max \frac{\|\Delta \mathbf{b}_i\|}{\|\mathbf{b}_i\|} \leq 2^{-\Omega(n)}$,
  then $B + \Delta B$ is reduced (for slightly weaker parameters).

Properties of the new reduction

The new reduction is perturbation-friendly:

- We still have $\mathrm{cond}(R) = 2^{\mathcal{O}(n)}$ for Ξ-reduced bases.
- If B is reduced and $\max \frac{\|\Delta \mathbf{b}_i\|}{\|\mathbf{b}_i\|} \leq 2^{-\Omega(n)}$, then $B + \Delta B$ is reduced (for slightly weaker parameters).

The popular properties of LLL-reduction still hold:

- Computable in polynomial time.
- B reduced $\implies \prod \|\mathbf{b}_i\| \leq 2^{\mathcal{O}(n^2)} \cdot |\det(\mathbf{b}_i)_i|$.

Deformations and truncations are compatible

- B and $\sigma_\ell BU$ reduced $\implies$ U small.
- B reduced $\implies$ $B + \Delta B$ reduced.

Let $\ell \geq 0$, B be reduced and ΔB s.t. $\max \frac{\|\Delta \mathbf{b}_i\|}{\|\mathbf{b}_i\|} \leq 2^{-\ell-\Omega(n)}$.
If $\sigma_\ell(B + \Delta B)U$ is reduced, then so is $\sigma_\ell BU$...
For slightly weaker reduction factors.

The $\ell + O(n)$ top-most bits of B suffice for finding U.

Deformations and truncations are compatible

- B and $\sigma_\ell BU$ reduced $\implies$ U small.
- B reduced $\implies$ $B + \Delta B$ reduced.

Let $\ell \geq 0$, B be reduced and ΔB s.t. $\max \frac{\|\Delta \mathbf{b}_i\|}{\|\mathbf{b}_i\|} \leq 2^{-\ell-\Omega(n)}$.
If $\sigma_\ell(B + \Delta B)U$ is reduced, then so is $\sigma_\ell BU$...
For slightly weaker reduction factors.

The $\ell + O(n)$ top-most bits of B suffice for finding U.

Deformations and truncations are compatible

- B and $\sigma_\ell BU$ reduced $\implies$ U small.
- B reduced $\implies$ $B + \Delta B$ reduced.

Let $\ell \geq 0$, B be reduced and ΔB s.t. $\max \frac{\|\Delta \mathbf{b}_i\|}{\|\mathbf{b}_i\|} \leq 2^{-\ell-\Omega(n)}$.
If $\sigma_\ell(B + \Delta B)U$ is reduced, then so is $\sigma_\ell BU$...
For slightly weaker reduction factors.

The $\ell + O(n)$ top-most bits of B suffice for finding U.

Deformations and truncations are compatible

- B and $\sigma_\ell BU$ reduced $\implies$ U small.
- B reduced $\implies$ $B + \Delta B$ reduced.

Let $\ell \geq 0$, B be reduced and ΔB s.t. $\max \frac{\|\Delta \mathbf{b}_i\|}{\|\mathbf{b}_i\|} \leq 2^{-\ell-\Omega(n)}$.
If $\sigma_\ell(B + \Delta B)U$ is reduced, then so is $\sigma_\ell BU$...
For slightly weaker reduction factors.

The $\ell + O(n)$ top-most bits of B suffice for finding U.

Plan of the talk

1. Wishful thinking.
2. Reducing by deforming.
3. Reducing by truncating.
4. **The $\widetilde{\mathbf{L}}^1$ algorithm.**

Overview of $\widetilde{L}^1$

- $\widetilde{L}^1$: HNF and n calls to `Lift-`$\widetilde{L}^1$.
- If B is reduced and $\ell \geq 0$, `Lift-`$\widetilde{L}^1$ computes U unimodular such that $\sigma_\ell BU$ is reduced, in time $\mathcal{P}oly(n) \cdot \widetilde{O}(\ell)$.

- We master "remainders/bases" truncations.
- We have "LLL quotients".
- If the basis is balanced, the quotient has small bit-size.

Overview of $\widetilde{\mathrm{L}}^1$

- $\widetilde{\mathrm{L}}^1$: HNF and n calls to $\texttt{Lift-}\widetilde{\mathrm{L}}^1$.
- If B is reduced and $\ell \geq 0$, $\texttt{Lift-}\widetilde{\mathrm{L}}^1$ computes U unimodular such that $\sigma_\ell BU$ is reduced, in time $\mathcal{P}oly(n) \cdot \widetilde{O}(\ell)$.

- We master "remainders/bases" truncations.
- We have "LLL quotients".
- If the basis is balanced, the quotient has small bit-size.

A first attempt for Lift-$\widetilde{L}^1$

Inputs: B reduced, lifting target ℓ.
Output: U unimodular such that $\sigma_\ell BU$ reduced.

- Keep the $\ell/2 + O(n)$ top-most bits of B.
- Recursively compute U_1 s.t. $\sigma_{\ell/2} BU_1$ reduced.
- Apply U_1 to $\sigma_{\ell/2} B$ and keep the $\ell/2 + O(n)$ top-most bits.
- Recursively compute U_2 s.t. $\sigma_{\ell/2}(\sigma_{\ell/2} BU_1)U_2$ is reduced.
- Return $U_1 \cdot U_2$.

A first attempt for Lift-$\widetilde{L}^1$

Inputs: B reduced, lifting target ℓ.
Output: U unimodular such that $\sigma_\ell BU$ reduced.

- Keep the $\ell/2 + O(n)$ top-most bits of B.
- Recursively compute U_1 s.t. $\sigma_{\ell/2}BU_1$ reduced.
- Apply U_1 to $\sigma_{\ell/2}B$ and keep the $\ell/2 + O(n)$ top-most bits.
- Recursively compute U_2 s.t. $\sigma_{\ell/2}(\sigma_{\ell/2}BU_1)U_2$ is reduced.
- Return $U_1 \cdot U_2$.

Some additional difficulties

1. Keep the $\ell/2 + O(n)$ top-most bits of B.
2. Recursively compute U_1 s.t. $\sigma_{\ell/2} B U_1$ reduced.
3. Apply U_1 to $\sigma_{\ell/2} B$ and keep the $\ell/2 + O(n)$ top-most bits.
4. Recursively compute U_2 s.t. $\sigma_{\ell/2}(\sigma_{\ell/2} B U_1) U_2$ is reduced.
5. Return $U_1 \cdot U_2$.

- What do we do at the recursion leaves?
- Every time we truncate, we may loosen the reduction factors...
- How do we compute $B \cdot U_1$ and $U_1 \cdot U_2$ efficiently?

Strengthening the reducedness of a basis (Morel-S-Villard)

Problem: Suppose we have a Ξ-reduced basis.
How do we Ξ'-reduce it, for $\Xi' > \Xi$?

- Truncate, reduce, output the obtained U.
- This takes time $\mathcal{O}(n^{6+\varepsilon})$ when the r_{ii}'s are balanced.

Otherwise, u_{ij} can be as large as r_{jj}/r_{ii} ...

Strengthening the reducedness of a basis (Morel-S-Villard)

Problem: Suppose we have a Ξ-reduced basis.
How do we Ξ'-reduce it, for $\Xi' > \Xi$?

- Truncate, reduce, output the obtained U.
- This takes time $\mathcal{O}(n^{6+\varepsilon})$ when the r_{ii}'s are balanced.

Otherwise, u_{ij} can be as large as r_{jj}/r_{ii} ...

Strengthening the reducedness of a basis (Morel-S-Villard)

Problem: Suppose we have a Ξ-reduced basis.
How do we Ξ'-reduce it, for $\Xi' > \Xi$?

- Truncate, reduce, output the obtained U.
- This takes time $\mathcal{O}(n^{6+\varepsilon})$ when the r_{ii}'s are balanced.

Otherwise, u_{ij} can be as large as r_{jj}/r_{ii} ...

Strengthening the reducedness of a basis (Morel-S-Villard)

1. Rescale the columns of B: $B \mapsto BS$.
2. Do that while keeping B reduced.
3. Find U unimodular s.t. $(BS)U$ is reduced.
4. $(BSU)S^{-1} = B(SUS^{-1})$ is reduced.
5. If the scaling was properly chosen: SUS^{-1} is unimodular.

- This costs $\mathcal{O}(n^{6+\varepsilon})$.
- It also works for a small amount of lift: $\ell = \mathcal{O}(n)$.

Strengthening the reducedness of a basis (Morel-S-Villard)

1. Rescale the columns of B: $B \mapsto BS$.
2. Do that while keeping B reduced.
3. Find U unimodular s.t. $(BS)U$ is reduced.
4. $(BSU)S^{-1} = B(SUS^{-1})$ is reduced.
5. If the scaling was properly chosen: SUS^{-1} is unimodular.

- This costs $\mathcal{O}(n^{6+\varepsilon})$.
- It also works for a small amount of lift: $\ell = \mathcal{O}(n)$.

Strengthening the reducedness of a basis (Morel-S-Villard)

1. Rescale the columns of B: $B \mapsto BS$.
2. Do that while keeping B reduced.
3. Find U unimodular s.t. $(BS)U$ is reduced.
4. $(BSU)S^{-1} = B(SUS^{-1})$ is reduced.
5. If the scaling was properly chosen: SUS^{-1} is unimodular.

- This costs $\mathcal{O}(n^{6+\varepsilon})$.
- It also works for a small amount of lift: $\ell = \mathcal{O}(n)$.

Strengthening the reducedness of a basis (Morel-S-Villard)

1. Rescale the columns of B: $B \mapsto BS$.
2. Do that while keeping B reduced.
3. Find U unimodular s.t. $(BS)U$ is reduced.
4. $(BSU)S^{-1} = B(SUS^{-1})$ is reduced.
5. If the scaling was properly chosen: SUS^{-1} is unimodular.

- This costs $\mathcal{O}(n^{6+\varepsilon})$.
- It also works for a small amount of lift: $\ell = \mathcal{O}(n)$.

Reducedness strengthening

- Used for the recursion leaves.
- Used for re-strengthening the reduction factors, loosened by the truncations.
- Returns (U, S) s.t.:
  - $B(SUS^{-1})$ is reduced,
  - $\max |u_{ij}| \leq 2^{O(n)}$,
  - S is powers-of-2 diagonal matrix.

SUS^{-1} might not be small, but (S, U) is.

Reducedness strengthening

- Used for the recursion leaves.
- Used for re-strengthening the reduction factors, loosened by the truncations.
- Returns (U, S) s.t.:
  - $B(SUS^{-1})$ is reduced,
  - $\max |u_{ij}| \leq 2^{O(n)}$,
  - S is powers-of-2 diagonal matrix.

SUS^{-1} might not be small, but (S, U) is.

Bounding the cost of $\texttt{Lift-}\widetilde{\mathtt{L}}^1$

1. Keep the $\ell/2 + O(n)$ top-most bits of B.
2. Recursively compute U_1 s.t. $\sigma_{\ell/2} B U_1$ reduced.
3. Apply U_1 to $\sigma_{\ell/2} B$ and keep the $\ell/2 + O(n)$ top-most bits.
4. Recursively compute U_2 s.t. $\sigma_{\ell/2}(\sigma_{\ell/2} B U_1) U_2$ is reduced.
5. Return $U_1 \cdot U_2$.

New representations for bases and transforms:

- Easy if assuming all handled bases are "balanced". Else...
- An ℓ-lifing U is stored as (U', D) with $U = DU'D^{-1}$, $\max |u'_{ij}| \leq 2^{\ell + \mathcal{O}(n)}$ and D p-of-2 diagonal.
- B is stored as (B', D) with $B = B'D$ and $\max |b'_{i,j}| \leq 2^{\mathcal{O}(\ell + n)}$ and D p-of-2 diagonal.

Bounding the cost of Lift-$\widetilde{\mathrm{L}}^1$

1. Keep the $\ell/2 + O(n)$ top-most bits of B.
2. Recursively compute U_1 s.t. $\sigma_{\ell/2} B U_1$ reduced.
3. Apply U_1 to $\sigma_{\ell/2} B$ and keep the $\ell/2 + O(n)$ top-most bits.
4. Recursively compute U_2 s.t. $\sigma_{\ell/2}(\sigma_{\ell/2} B U_1) U_2$ is reduced.
5. Return $U_1 \cdot U_2$.

New representations for bases and transforms:

- Easy if assuming all handled bases are "balanced". Else...
- An ℓ-lifing U is stored as (U', D) with $U = DU'D^{-1}$, $\max |u'_{ij}| \leq 2^{\ell + \mathcal{O}(n)}$ and D p-of-2 diagonal.
- B is stored as (B', D) with $B = B'D$ and $\max |b'_{i,j}| \leq 2^{\mathcal{O}(\ell+n)}$ and D p-of-2 diagonal.

Bounding the cost of Lift-$\widetilde{\mathrm{L}}^1$

1. Keep the $\ell/2 + O(n)$ top-most bits of B.
2. Recursively compute U_1 s.t. $\sigma_{\ell/2} B U_1$ reduced.
3. Apply U_1 to $\sigma_{\ell/2} B$ and keep the $\ell/2 + O(n)$ top-most bits.
4. Recursively compute U_2 s.t. $\sigma_{\ell/2}(\sigma_{\ell/2} B U_1) U_2$ is reduced.
5. Return $U_1 \cdot U_2$.

New representations for bases and transforms:

- Easy if assuming all handled bases are “balanced”. Else...
- An ℓ-lifing U is stored as (U', D) with $U = DU'D^{-1}$, $\max |u'_{ij}| \leq 2^{\ell+\mathcal{O}(n)}$ and D p-of-2 diagonal.
- B is stored as (B', D) with $B = B'D$ and $\max |b'_{i,j}| \leq 2^{\mathcal{O}(\ell+n)}$ and D p-of-2 diagonal.

Handling the new representations

- $U \mapsto (U', D)$ with $U = DU'D^{-1}$.
- $B \mapsto (B', D)$ with $B = B'D$.

- $(B_1 D_1) \cdot (D_2 U_2 D_2^{-1})$ is cheap if D_1^{-1} and D_2 "coincide".
- $(D_1 U_1 D_1^{-1}) \cdot (D_2 U_2 D_2^{-1})$ is cheap if D_1 and D_2 "coincide".
- They always do coincide: $D \approx \mathrm{diag}(r_{11}, \ldots, r_{nn})$.

Final hassle:

- The bit-sizes of the DUD^{-1}'s might grow too much.
- We sanatize them at every recursion leaf.

Handling the new representations

- $U \mapsto (U', D)$ with $U = DU'D^{-1}$.
- $B \mapsto (B', D)$ with $B = B'D$.

- $(B_1 D_1) \cdot (D_2 U_2 D_2^{-1})$ is cheap if D_1^{-1} and D_2 "coincide".
- $(D_1 U_1 D_1^{-1}) \cdot (D_2 U_2 D_2^{-1})$ is cheap if D_1 and D_2 "coincide".
- They always do coincide: $D \approx \mathrm{diag}(r_{11}, \ldots, r_{nn})$.

Final hassle:

- The bit-sizes of the DUD^{-1}'s might grow too much.
- We sanatize them at every recursion leaf.

Handling the new representations

- $U \mapsto (U', D)$ with $U = DU'D^{-1}$.
- $B \mapsto (B', D)$ with $B = B'D$.

- $(B_1 D_1) \cdot (D_2 U_2 D_2^{-1})$ is cheap if D_1^{-1} and D_2 "coincide".
- $(D_1 U_1 D_1^{-1}) \cdot (D_2 U_2 D_2^{-1})$ is cheap if D_1 and D_2 "coincide".
- They always do coincide: $D \approx \mathrm{diag}(r_{11}, \ldots, r_{nn})$.

Final hassle:

- The bit-sizes of the DUD^{-1}'s might grow too much.
- We sanatize them at every recursion leaf.

Sanitizing the transforms

Assume B and $\sigma_\ell BU$ are reduced with $\ell \geq 0$ and U unimodular. Let ΔU s.t. $|\Delta u_{ij}| \leq 2^{-\Omega(\ell+n)} r_{jj}/r_{ii}$, then:

$$U + \Delta U \text{ unimodular} \quad \text{and} \quad \sigma_\ell B(U + \Delta U) \text{ reduced.}$$

- A lift-reducing U may be large, but its bit-size can be made small.
- To "clean" a DUD', we equalize D^{-1} and D', and truncate.
- $U \mapsto (U', D, x)$ with $U = 2^x DU'D^{-1}$.

Sanitizing the transforms

Assume B and $\sigma_\ell BU$ are reduced with $\ell \geq 0$ and U unimodular. Let ΔU s.t. $|\Delta u_{ij}| \leq 2^{-\Omega(\ell+n)} r_{jj}/r_{ii}$, then:

$$U + \Delta U \text{ unimodular} \quad \text{and} \quad \sigma_\ell B(U+\Delta U) \text{ reduced.}$$

- A lift-reducing U may be large, but its bit-size can be made small.
- To "clean" a DUD', we equalize D^{-1} and D', and truncate.
- $U \mapsto (U', D, x)$ with $U = 2^x DU'D^{-1}$.

Sanitizing the transforms

Assume B and $\sigma_\ell BU$ are reduced with $\ell \geq 0$ and U unimodular. Let ΔU s.t. $|\Delta u_{ij}| \leq 2^{-\Omega(\ell+n)} r_{jj}/r_{ii}$, then:

$$U + \Delta U \text{ unimodular and } \sigma_\ell B(U + \Delta U) \text{ reduced.}$$

- A lift-reducing U may be large, but its bit-size can be made small.
- To "clean" a DUD', we equalize D^{-1} and D', and truncate.
- $U \mapsto (U', D, x)$ with $U = 2^x DU'D^{-1}$.

Sanitizing the transforms

Assume B and $\sigma_\ell BU$ are reduced with $\ell \geq 0$ and U unimodular. Let ΔU s.t. $|\Delta u_{ij}| \leq 2^{-\Omega(\ell+n)} r_{jj}/r_{ii}$, then:

$$U + \Delta U \text{ unimodular and } \sigma_\ell B(U + \Delta U) \text{ reduced.}$$

- A lift-reducing U may be large, but its bit-size can be made small.
- To "clean" a DUD', we equalize D^{-1} and D', and truncate.
- $U \mapsto (U', D, x)$ with $U = 2^x DU'D^{-1}$.

Conclusion and open problems

- $\widetilde{L}^1$ reduces in time $\mathcal{O}(n^{5+\varepsilon}\beta + n^{\omega+1+\varepsilon}\beta^{1+\varepsilon})$.
- This generalizes Knuth-Schönhage and Schönhage-Yap to arbitrary dimensions.
- Three ingredients: deforming, truncating, Knuth-Schönhage.

1. Can we do better wrt n? [Schönhage'84, Storjohann'96, etc]
2. How does it compare to BKZ_2? [Hanrot-Pujol-S'11]
3. Can we use these techniques for other objects?

Conclusion and open problems

- $\widetilde{\mathrm{L}}^1$ reduces in time $\mathcal{O}(n^{5+\varepsilon}\beta + n^{\omega+1+\varepsilon}\beta^{1+\varepsilon})$.
- This generalizes Knuth-Schönhage and Schönhage-Yap to arbitrary dimensions.
- Three ingredients: deforming, truncating, Knuth-Schönhage.

1. Can we do better wrt n? [Schönhage'84, Storjohann'96, etc]
2. How does it compare to BKZ_2? [Hanrot-Pujol-S'11]
3. Can we use these techniques for other objects?