The New LLL Routine in the MAGMA Computational Algebra System

Damien Stehlé

`http://www.loria.fr/~stehle`

University of Sydney/Université Nancy 1

Plan of the Talk

- Reminders on the LLL algorithm.
- How to use the new MAGMA LLL.
- NTL versus MAGMA.
- Further improvements.

Quick bibliography

- Lenstra-Lenstra-Lovász ’82.
- Schnorr-Euchner ’94.
- Nguyễn-Stehlé ’05.
- Nguyễn-Stehlé ’06.

1) Reminders on lattices

Lattices are useful in many areas

- Computer algebra.
- Algorithmic number theory.
- Computational group theory.
- Linear integral relation detection.
- Cryptanalysis.
- Computer arithmetic.

Euclidean lattices

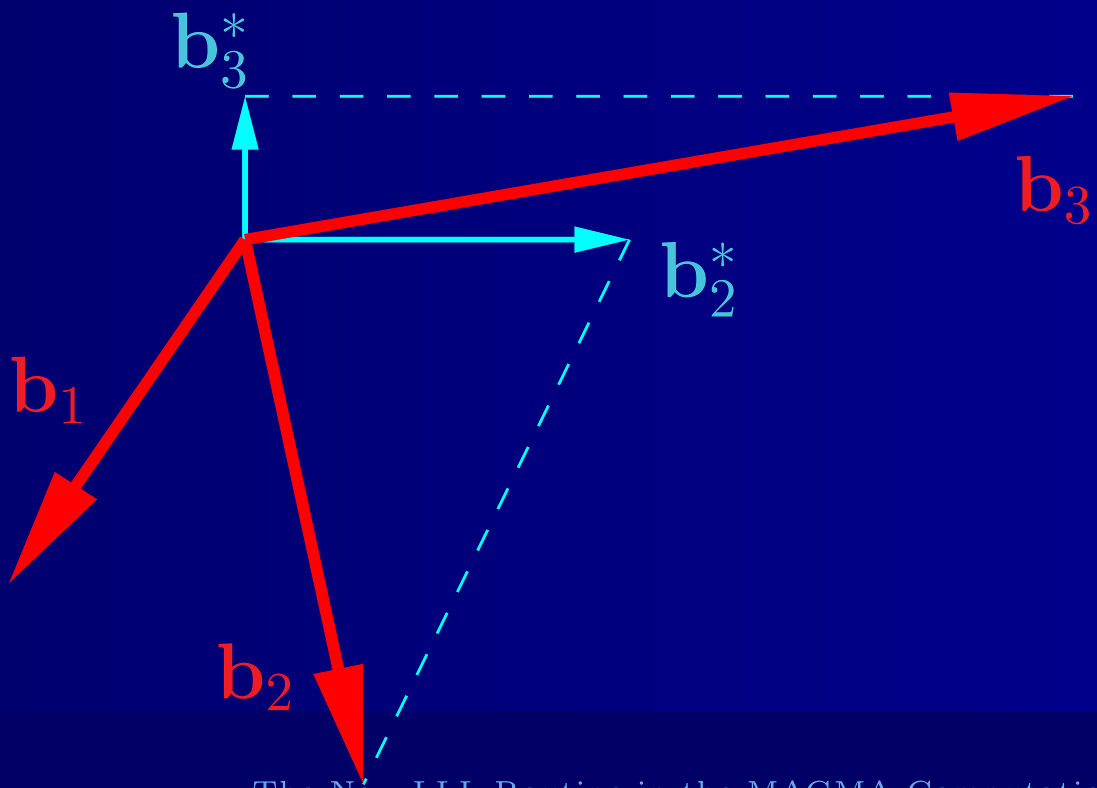
- Lattice = discrete subgroup of $\mathbb{R}^n$.
- $L[\mathbf{b}_i] = \left\{ \sum_{i=1}^{d} x_i \mathbf{b}_i, x_i \in \mathbb{Z} \right\}$,
  represented by a $d \times n$ matrix.
- If the b_i's are linearly independent, they are a lattice basis.
- Dimension: shared cardinality of the bases.
- First minimum: $\lambda(L) = \min(\|\mathbf{b}\| : \mathbf{b} \in L \backslash \{0\})$.

In this talk, $d = n$ and $L \subset \mathbb{Z}^d$.

Gram-Schmidt orthogonalisation

$\mathbf{b}_1^* = \mathbf{b}_1, \;\; \mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j}\mathbf{b}_j^*, \;\; \mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\|\mathbf{b}_j^*\|^2}$.

$\det L[\mathbf{b}_i] = \prod \|\mathbf{b}_i^*\|$ is independent of the $\mathbf{b}_i$'s.



Two main computational tasks

Given a basis of a d-dimensional lattice L, compute a vector $\mathbf{b}_1 \in L$ which is:

- not very long as regard to $\lambda(L)$.
- not very long as regard to $\det(L)^{1/d}$.

LLL reduction

- A basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ is (δ, η)-LLL-reduced if:
  (1) $\forall i > j, \quad |\mu_{i,j}| \leq \eta$.
  (2) $\forall i, \quad \delta \cdot \|\mathbf{b}_{i-1}^*\| \leq \|\mathbf{b}_i^* + \mu_{i,i-1}\mathbf{b}_{i-1}^*\|$,
  where $\delta \in (0.25, 1)$ and $\eta \in (0.5, \sqrt{\delta})$.
- (2) means: in $(\mathbf{b}_1, \ldots, \mathbf{b}_{i-2})^\perp$, $\mathbf{b}_{i-1}$ is approx. shorter than $\mathbf{b}_i$.
- Often $(\delta, \eta) = (0.999, 0.501)$.

Properties of LLL-reduced bases

- $\|b_1\| \leq \left(\delta - \eta^2\right)^{-(d-1)/4} \cdot (\det L)^{1/d}$,
- $\|b_1\| \leq (\delta - \eta^2)^{-(d-1)/2} \cdot \lambda(L)$,

- $\prod_{i=1}^{d} \|b_i\| \leq (\delta - \eta^2)^{-d(d-1)/4} \cdot (\det L)$,
- $\forall j < i,\ \|b_j^*\| \leq (\delta - \eta^2)^{(j-i)/2} \cdot \|b_i^*\|$.

The rational LLL algorithm

Input: $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ linearly independent.

Output: A LLL-reduced basis of $L[\mathbf{b}_i]$.

1. $\kappa := 2$. While $\kappa \leq d$, do:
2. Make all the $|\mu_{\kappa,i}|$'s smaller than η:
3. Compute the $\mu_{\kappa,i}$'s.
4. For i from $\kappa - 1$ down to 1 do, if $|\mu_{\kappa,i}| \geq \eta$:
5. $\mathbf{b}_\kappa := \mathbf{b}_\kappa - \lfloor \mu_{\kappa,i} \rceil \mathbf{b}_i$.
6. For j from 1 to i do $\mu_{\kappa,j} := \mu_{\kappa,j} - \lfloor \mu_{\kappa,i} \rceil \mu_{i,j}$.
7. If $\delta \|\mathbf{b}_{\kappa-1}^*\| \leq \|\mathbf{b}_\kappa^* + \mu_{\kappa,\kappa-1}\mathbf{b}_{\kappa-1}^*\|$, then $\kappa := \kappa + 1$.
8. Else swap $\mathbf{b}_{\kappa-1}$ and $\mathbf{b}_\kappa$, $\kappa := \max(\kappa - 1, 2)$.

The floating-point LLL

- Classical LLL: Gram-Schmidt computations done with rational numbers with huge numerators and denominators.
- *fp*-LLL: Gram-Schmidt computations done with floating-point approximations with much smaller mantissas.
- To get a provable *fp*-LLL, one needs arbitrary precision *fp* numbers and the Gram matrix of the basis [Stehlé-Nguyễn '05].

LLL implementations

Fast LLL implementations rely on floating-point computations, based on [Schnorr-Euchner '94].

- NTL.
- MAGMA.
- Pari GP.
- LiDIA.
- Maple, Mathematica, Gap.

2) The new LLL routine of MAGMA

Main properties

- Correctness.
- Termination.
- Reasonably fast
  (in particular with the `Fast` option).
- Works for linearly dependent vectors and
  all symmetric matrices.

Correctness

When `Proof` is `true`, the output basis is (δ, η)-LLL-reduced.
MAGMA contains the only guaranteed *fp*-LLL.

- Internally, δ and η are strengthened.
- The output is not sorted by length anymore.

To obtain better timings than before, set `Proof` to `false`, or use `LatticeReduce`.

Main options

Warning:
the default variant is seldom the one you want.

- LLL parameters δ and η (default: 0.75, 0.501).
- `SwapCondition`. Siegel's condition:

$$\|\mathbf{b}_{i+1}^*\|^2 \geq (\delta - \eta^2) \cdot \|\mathbf{b}_i^*\|^2.$$

- `EarlyReduction`. Vectors can be size-reduced in advance.
- `Fast`. The above parameters are chosen automatically for you.

You want a LLL-reduced basis

- Keep the default variant.
- Eventually set (δ, η) closer to $(1, 1/2)$.
- Eventually set `Proof` to `false`.

You want the main LLL properties

- Set `SwapCondition` to `Siegel`.
- Eventually set `EarlyReduction` to `true`.
- Eventually set `Proof` to `false`.

You want a somehow reduced basis

- Set `Proof` to `false`.
- Activate the `Fast` option.
  It will output a LLL-reduced basis for some factors δ, η.
- These factors will be given to you.

3) Comparison of diverse LLLs

Compared software

- MAGMA 2.12 and 2.13, NTL 5.4.
- On a 2.4 GHz AMD Opteron.
- Using GNU MP 4.2.1 and Gaudry's patch, for both NTL and MAGMA.
- Using MPFR 2.2.0 for MAGMA.
- All timings in seconds.
- $\delta = 0.75, \eta = 0.501$.

Termination test in dimension 3

$$\begin{bmatrix} 1 & -1 & 0 \\ 2^{100}+1 & 2^{100}-1 & 0 \\ 2^{100} & 2^{100}-1 & 1 \end{bmatrix}.$$

- NTL's `G_LLL_FP` loops forever.
- MAGMA 2.12's LLL without `UseGram` and `UnderflowCheck`: falls down to integral method.
- PARI: incorrect answer (2 weeks ago).

Termination test in dimension 55

Worst-case for the correctness proof of [Nguyễn-Stehlé ’05].

- NTL’s `LLL_FP` and `LLL_XD` loop forever.
- MAGMA 2.12’s LLL without `UseGram`: falls down to integral method. 3.43s.
- PARI: 2.67s.
- MAGMA 2.13’s LLL: 0.014s.

Uniform entries in dimension 1000

All entries uniformly chosen with $\log B$ bits.

$\log B$	NTL	MAGMA 2.12	MAGMA 2.13
10	5.43	6.03	5.49
1000	204	46.8	13.2

Pari: $> 8000s$ for the first matrix.

Knapsack-type bases

Non trivial entries are $\log B$ bit long.

d	$\log B$	NTL	V2.12	V2.13	V2.13 `Fast`
10	$100,000$	37.6	6.69	5.16	2.99
100	$10,000$	344	269	134	42.1
150	$5,000$	$\infty^1, 3240$	4993	597	250

[1]NTL's `LLL_XD` loops forever: $\Rightarrow$ `LLL_RR`.

Simult. Diophantine approximation

Dimension 76, non-trivial entries of ≈ 5000 bits.

NTL	V2.12	V2.13	V2.13 `Fast`
1142	$\infty?$	76.5	42.8

4) Further improvements

Possible improvements for LLL

- Givens and Householder orthogonalisations?
- Provable variant without the Gram matrix. Technical difficulty: computing a portion of the product of two integers.
- Low-level improvement of the integer operation “big $+$ small $\times$ big”.

Other LLL-related routines

- `PowerRelations` and `IntegerRelations`.
- Coppersmith's method for the small roots of polynomials (the modular univariate case is already available).
- Schnorr's block-Korkine-Zolotarev algorithm.