A Flatter implementation in PARI

B. Allombert

IMB
CNRS/Université de Bordeaux

12/01/2024

FLATTER

FLATTER is a new lattice reduction algorithm developped by Keegan Ryan and Nadia Heninger which is much faster for a lot of instance that are relevant to number theory and PARI/GP, in particular, T_2 reduction of order and ideal bases, and knapsack-like lattice of small dimensions with large entries.
Keegan Ryan provides a fast implementation targetting cryptographic challenges.
We implemented in PARI/GP a version of the algorithm more suitable for applications in number theory.

Lattices

For the purpose of reduction, a n-dimensional lattice is a $\mathbb{Z}$-linear map from $\mathbb{Z}^n$ to $(\mathbb{R}^n, <>)$ with matrix M.
There are two kinds of basis changes:

- Right basis change by a matrix in $GL_n(\mathbb{Z})$,
- Left basis change by a matrix in $O_n(\mathbb{R})$.

Only the right basis change actually reduce the lattice.

Gram-Schmidt orthogonalization

Gram-Schmidt orthogonalization computes an orthogonal matrix U such that UM is upper-triangular with non-zeros diagonal entries. Unfortunately computing U using exact arithmetic is slow, and using floating point approximation can lead to accuracy errors.

Profile

The diagonal entries $b^*_{i,i}$ of UM (the norms of the Gram-Schmidt basis vectors) allows to compute the profile which allows to measure the quality of the reduction, in particular the potential $\prod ||b^*_{i,i}||^{1/n+1-i}$ and the spread $\max(||b^*_{i,i}||)/min(||b^*_{i,i}||)$. and the drop $D = \sum_{i=1}^{n-1} \max(0, \log(|b^*_{i,i}/b^*_{i+1,i+1}|))$.
Ryan-Heninger suggest to use as bit precision $30 + 3n + 2D$.
However, to compute the drop, we need enough precision so that the resulting $||b^*_{i,i}||$ are not too inaccurate.

Integral renormalization

If a matrix M has floating point entries, before performing LLL, the matrix is rescaled by a power of 2 depending of the accuracy and the exponents and entries are truncated to integers.

FLATTER algorithm

1. If M is not integral, perform Integral renormalization on M.
2. Apply the following recursive reduction step until the matrix is close to reduced:
    2.1 Gram-Schmidt orthogonalization
    2.2 First Recursions
    2.3 Size reduction
    2.4 Second Gram-Schmidt orthogonalization
    2.5 Second recursion
3. Perform LLL reduction on the nearly reduced matrix and return.

Step 1: Gram-Schmidt orthogonalization

Perform Gram-Schmidt orthogonalization on M using floating point arithmetic to some precision to find an orthogonal matrix U so that $R = UM$ is upper triangular. (Practically we only need to compute R).

Step 2: First Recursions

Write R as $R = \begin{pmatrix} R_1 & R_2 \\ 0 & R_3 \end{pmatrix}$, with R_1 of dimension $[d/2]$.
Note that R_1 and R_3 are upper triangular.
Call recursively the algorithm to R_1 and R_3 to get unimodular transformation matrices T_1 and T_3 and set $T = \begin{pmatrix} T_1 & 0 \\ 0 & T_3 \end{pmatrix}$ which is also unimodular.

Step 3: Size reduction

Set $S = \begin{pmatrix} 1 & S_2 \\ 0 & 1 \end{pmatrix}$ for some integral matrix S_2.

$$RTS = \begin{pmatrix} R_1 T_1 & R_1 T_1 S_2 + R_3 T_3 \\ 0 & R_3 T_3 \end{pmatrix} .$$

Pick $S_2 = \mathrm{round}(T_1^{-1}(R_1^{-1} R_3) T_3)$ so as to minimize $R_1 T_1 S_2 + R_3 T_3$.

$$TS = \begin{pmatrix} T_1 & T_1 S_2 \\ 0 & T_3 \end{pmatrix} .$$

Step 4: Second Gram-Schmidt orthogonalization

Perform Gram-Schmidt orthogonalization on MTS using floating point arithmetic to some precision to find an orthogonal matrix U' so that $R' = U'MTS$ is upper triangular. (Practically we only need to compute R).

Step 5: Second recursion

Write R' as

$$R' = \begin{pmatrix} R'_1 & \dots & \dots \\ 0 & R'_2 & \dots \\ 0 & 0 & R'_3 \end{pmatrix}$$

with R'_2 of size $[d/2]$ and R'_1 of size $[d/4]$. Note that R'_1, R'_2 and R'_3 are upper triangular and R'_1 and R'_3 are already reduced. Call recursively the algorithm to reduce a rescaled integral form of R'_2 to get an unimodular transformation matrix T'_2. Set T' to

$$T' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & T'_2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

and returns TST'.

The FLATTERGRAM algorithm (by Bill)

FLATTERGRAM is a variant of FLATTER for lattice given by an integral Gram matrix G.

1. Apply the following reduction step until the matrix is close to reduced:
    1.1 Apply Cholesky algorithm using floating point arithmetic to some precision to find M such that ${}^tMM = G$.
    1.2 Apply FLATTER to M to get an unimodular transformation matrix T and replaces G by ${}^tTGT$.

The FLATTERKER algorithm (by Aurel)

FLATTERKER is a variant of FLATTER for matrices that are not of maximal rank.
Do the following with $i = 1, 2, \ldots$ until M is reduced.

1. apply FLATTER to $\begin{pmatrix} 2^i M \\ I_n \end{pmatrix}$ to get an unimodular transformation matrix T and replace M by MT.

Tuning

At each reduction step it is necessary to decide whether to use fplll or FLATTER. We have different tuning for generic lattices and knapsack. We currently use the spread for estimating the cost of LLL. We should experiment with using the potential.