Complexity of an extended lattice reduction algorithm

Wilberd van der Kallen

December 14, 1998

1 Summary

We consider the complexity of a Lenstra Lenstra Lovász lattice reduction algorithm ([LLL]) in which the vectors are allowed to be linearly dependent and in which one also asks for the matrix of the transformation from the given generators to the reduced basis. The main problem will be to show that the entries of the transformation matrix remain bounded through the algorithm, with a reasonable bound. Here the difficulty is of course that due to the dependence of the generators the transformation is not determined by the basis. To remedy this we work with two inner products and apply the LLL methods to both.

2 Description of `GramLatticeReduce`

Let $e_1,\ldots,e_n$ be the standard basis of $\mathbb{R}^n$. The input of `GramLatticeReduce` is the Gram matrix $gram = (\langle e_i, e_j\rangle)_{i,j=1}^n$ of a positive semidefinite inner product $\langle\ ,\ \rangle$ on $\mathbb{R}^n$. We assume *gram* has integer entries. We are concerned with the lattice $\mathbb{Z}^n$. The output of `GramLatticeReduce` is an integer *rank* and an integer matrix b of determinant one. To explain its properties we need some more notation. The ordinary inner product on $\mathbb{R}^n$ is denoted $(\ ,\ )$. Call v isotropic if $\langle v, v\rangle = 0$. Put $isodim = n - rank$ and let b_i^* denote the i-th Gram-Schmidt vector in the following sense. We have $b_i^* \in (b_i + \sum_{j=1}^{i-1}\mathbb{R}b_i)$ and if $1 \le j < i$, $j \le isodim$ then $(b_i^*, b_j) = 0$, but if $1 \le j < i$, $j > isodim$ then $\langle b_i^*, b_j\rangle = 0$. With those notations the output satisfies:

1. The first *isodim* rows b_i of b are isotropic.

2. With respect to $(\ ,\ )$ the first *isodim* rows of b form an LLL reduced basis of $\sum_{j=1}^{isodim} \mathbb{Z}b_i$.

3. With respect to $\langle\ ,\ \rangle$ the last *rank* rows of b form an LLL reduced basis of the lattice they span, and this lattice contains no nonzero isotropic vector.

4. For $1 \le i \le isodim$, $i < j \le n$ we have $|(b_i^*, b_j)| \le 1/2(b_i^*, b_i)$.

Remark 2.1 Variations are possible, depending on what one is really after. If one is only interested in the isotropic vectors, one may weaken condition 3 to

3' The last *rank* rows of b form a basis of the lattice they span, and this lattice contains no nonzero isotropic vector. Furthermore, $|\langle b_i^*, b_j\rangle| \le 1/2\langle b_i^*, b_i\rangle$ for $n - rank + 1 \le i \le n$, $i < j \le n$.

Similarly, one may wish to replaces condition 2 with

2' For $1 \le i \le isodim$, $i < j \le isodim$ we have $|(b_i^*, b_j)| \le 1/2(b_i^*, b_i)$.

These changes do not affect our analysis in any essential way. One just has to change the wording, not the formulas. And in the algorithm one has to leave out some swaps.

3 Description of `ExtendedLatticeReduce`

Given generators $b_1, \ldots, b_n$ of a sublattice of $\mathbb{Z}^m$, the algorithm `ExtendedLatticeReduce` basically just calls `GramLatticeReduce` with as input the Gram matrix $((b_i, b_j))_{i,j=1}^n$.

4 Sketch of the algorithm

We assume the reader is familiar with [LLL] and also with the implementation of the LLL algorithm in integer arithmetic, as described in [C].

Most of the time we are given

- An integer matrix b of determinant one,

- Integers k, $kmax$, $1 \leq k \leq kmax \leq n$,
- An integer $isodim \geq 0$, so that the first $isodim$ rows b_i of b span the isotropic subspace of $\sum_{j=1}^{kmax} \mathbb{R} b_j$.

(Initialize with $k = kmax = 1$ and $isodim = 0$.)

Let $\mathrm{pr}_{\mathrm{iso}}$ be the orthogonal projection according to $(\ ,\ )$ of $\mathbb{R}^n$ onto $\sum_{j=1}^{isodim} \mathbb{R} b_j$ and put

$$(v, w)_{\mathrm{mix}} = (\mathrm{pr}_{\mathrm{iso}} v, \mathrm{pr}_{\mathrm{iso}} w) + \langle v, w \rangle.$$

Let $\mu_{i,j}$ be defined for $i > j$ so that

$$b_i = b_i^* + \sum_{j=1}^{i-1} \mu_{i,j} b_j^*.$$

The first standard assumption is then that, with respect to $(\ ,\ )_{\mathrm{mix}}$, the first $k-1$ rows of b form an LLL reduced basis of $\sum_{j=1}^{k-1} \mathbb{Z} b_j$, except that one does *not* require

$$|b_i^* + \mu_{i,i-1} b_{i-1}^*|_{\mathrm{mix}}^2 \geq 3/4 |b_{i-1}^*|_{\mathrm{mix}}^2$$

when $i = isodim + 1$. And the second standard assumption is that, as in [C], the first $kmax$ rows of b form a basis of $\sum_{j=1}^{kmax} \mathbb{Z} e_i$.

We run the LLL algorithm with respect to $(\ ,\ )_{\mathrm{mix}}$, except that one never swaps b_{isodim} with $b_{isodim+1}$. This roughly amounts to running two LLL algorithms, one for $(\ ,\ )$ and one for $\langle\ ,\ \rangle$. That is how one implements it and that is how we would have told it if we had not needed $(\ ,\ )_{\mathrm{mix}}$ for the complexity analysis. One runs LLL until k tries to go to $kmax + 1$. If $kmax = n$ we are through. If $kmax < n$ and $\sum_{j=1}^{kmax+1} \mathbb{Z} b_j$ contains no more isotropic vectors than $\sum_{j=1}^{kmax} \mathbb{Z} b_j$, then we simply increase $kmax$ by one.

In the remaining case we have to work until we are back in the standard situation with $k = 2$ and with both $isodim$ and $kmax$ one bigger. This is what `trickledown` is for. We postpone its discussion. At the end of `trickledown` we also make that $|\mu_{i,j}| \leq 1/2$ for $j < i \leq kmax$. This may not be necessary (and indeed we did not do this in earlier versions) but it can do little harm and definitely simplifies the estimates below.

5 Estimates

We want to give estimates as in [LLL]. Thus let $B \geq 2$ so that the entries of *gram* are at most B. Our main result is that all through the algorithm all entries have bit length $\mathcal{O}(n \log(nB))$. We do not care about the constants in this estimate. We leave to the reader the easy task of estimating the number of operations in the algorithm in the manner of [LLL].

5.1 Determinants

Let $gram_{\mathrm{mix}}$ be the Gram matrix $((e_i, e_j)_{\mathrm{mix}})$ with respect to $e_1, \ldots, e_{kmax}$. Its entries are at most $B+1$. With Hadamard this gives

$$|\det(gram_{\mathrm{mix}})| \leq (\sqrt{n}(B+1))^n$$

and the same estimate holds for its subdeterminants. We claim that the determinant of $gram_{\mathrm{mix}}$ is an integer, so that we also get this upper bound for the entries of $gram_{\mathrm{mix}}^{-1}$. To see the claim, consider as in [P] the inner product $(\ ,\ )_\epsilon$ given by $(v,w)_\epsilon = \epsilon(v,w) + \langle v,w\rangle$. Its Gram matrix has a determinant which is a polynomial $\det_\epsilon$ of ϵ with integer coefficients. One may also compute $\det_\epsilon$ with respect to a basis which is obtained from e_1, $\ldots$, e_{kmax} through an orthogonal transformation matrix. By diagonalizing the Gram matrix of $\langle\ ,\ \rangle$ we see that $\det(gram_{\mathrm{mix}})$ is the coefficient of ϵ^{isodim} in $\det_\epsilon$. □

Lemma 5.2 *For $v \in \mathbb{R}^n$ one has*

$$(v,v)_{\mathrm{mix}} \leq n(B+1)(v,v)$$

and for $v \in \sum_{j=1}^{kmax} \mathbb{R}e_j$ one has

$$(v,v) \leq n(\sqrt{n}(B+1))^n (v,v)_{\mathrm{mix}}.$$

Proof

The supremum of $\{(v,v)_{\mathrm{mix}} \mid (v,v) = 1\}$ is the largest eigenvalue of the gram matrix of $(\ ,\ )_{\mathrm{mix}}$ with respect to $e_1, \ldots, e_n$. The largest eigenvalue is no larger than the trace of this matrix. So it is at most $n(B+1)$. Similarly the largest eigenvalue of $gram_{\mathrm{mix}}^{-1}$ it is at most $n(\sqrt{n}(B+1))^n$. □

5.3 Vectors

Now put

$$diso_i = \prod_{j=1}^{i} (b_j^*, b_j^*)$$

for $i \leq isodim$ and

$$d_i = \prod_{j=1}^{i} \langle b_{j+isodim}^*, b_{j+isodim}^* \rangle$$

for $i \leq rank$. As far as d_i is concerned we may compute modulo isotropic vectors, or also with $(\ ,\ )_{\mathrm{mix}}$. Indeed

$$\langle b_{j+isodim}^*, b_{j+isodim}^* \rangle = (b_{j+isodim}^*, b_{j+isodim}^*)_{\mathrm{mix}}$$

for $1 \leq j \leq rank$. Both $diso_i$ and d_j are integers and they descend when applying LLL.

One may also compute $\det(gram_{\mathrm{mix}})$ with the b_i^* basis, as the transition matrix has determinant one. From that one sees that it is just $diso_{isodim} d_{rank}$. So we get $diso_{isodim} \leq (\sqrt{n}(B+1))^n$. In fact, for $i \leq isodim$ one has the same estimate

$$diso_i \leq (\sqrt{n}(B+1))^n$$

because i was equal to $isodim$ earlier in the algorithm and LLL only makes it go down. Similarly $d_{rank} \leq (\sqrt{n}(B+1))^n$, but actually we know from [LLL] that

$$d_i \leq B^i.$$

(This is not spoiled by `trickledown` which also makes d_i descend.)

Lemma 5.4 *Let $1 \leq i \leq kmax$. Then*

$$(\sqrt{n}(B+1))^{-n} \leq (b_i^*, b_i^*)_{\mathrm{mix}} \leq (\sqrt{n}(B+1))^n$$

and if $|\mu_{ij}| \leq 1/2$ for $1 \leq j < i$ then

$$(b_i, b_i)_{\mathrm{mix}} \leq n(\sqrt{n}(B+1))^n$$

□

Remark 5.5 These estimates are not as sharp as those in [LLL] but that can not be helped: It is no longer true that $(b_1^*, b_1^*) \leq B$. Consider for instance the quadratic form $\langle v, v \rangle = \sum_{i=1}^{n-1} (Nx_i - x_{i+1})^2$ for some large integer N. The shortest nonzero isotropic vector in $\mathbb{Z}^n$ is $(1, N, N^2, \ldots, N^n)$.

5.6 Preserved estimates

Lemma 5.7 *The following estimates hold between applications of* `trickledown`.

1. $diso_i \leq (\sqrt{n}(B+1))^n$ *for* $i \leq isodim$,
2. $d_i \leq B^i$ *for* $i \leq rank$,
3. $(b_i, b_i)_{\mathrm{mix}} \leq n(\sqrt{n}(B+1))^n$ *for* $i \neq k$,
4. $(b_k, b_k)_{\mathrm{mix}} \leq n^2 4^n (\sqrt{n}(B+1))^{3n}$,
5. $|\mu_{i,j}| \leq 1/2$ *for* $1 \leq j < i < k$,
6. $|\mu_{k,j}| \leq 2^{n-k}\sqrt{n}(\sqrt{n}(B+1))^n$ *for* $1 \leq j < k$,
7. $|\mu_{i,j}| \leq \sqrt{n}(\sqrt{n}(B+1))^n$ *for* $1 \leq j < i > k$.

Proof

That these are preserved under LLL follows as in [LLL], so one has to check that they hold right after `trickledown`. Given our earlier estimates this is straightforward. □

6 Description of `trickledown`

Before we can do estimates concerning `trickledown` we must describe it. One starts with having $k = kmax + 1 \leq n$. Consider the lattice generated by $b_1, \ldots, b_{kmax+1}$ where $b_{kmax+1} = e_{kmax+1}$. By assumption this lattice contains a nonzero vector v with $(v, v)_{\mathrm{mix}} = 0$. Modulo $\mathbb{R}v$ the vector b_k is linearly dependent on the b_i with $i < k$. Changing the basis of $\mathbb{Z}b_{k-1} + \mathbb{Z}b_k$ we can achieve that modulo $\mathbb{R}v$ the vector b_{k-1} is linearly dependent on the b_i with $i < k-1$. Then lower k by one and repeat until $k = isodim + 1$, where $isodim$ is the one from before the present `trickledown`. At that point b_k is itself isotropic and we increase $isodim$ by one and pass to a new $(\ ,\ )_{\mathrm{mix}}$. After subtracting suitable multiples of b_j with $j < i$ from b_i for all i we arrive at the situation where $|\mu_{i,j}| \leq 1/2$ and we leave `trickledown` with $k = 2$ (or $k = \max(isodim, 2)$) and with $kmax$ increased by one.

7 Estimates during `trickledown`

We look in more detail. Upon entering `trickledown` we freeze the old *isodim*, *kmax* and the b_i^*, even though the b_i will change. We also do not change $(\ ,\ )_{\rm mix}$. Let $\mu_{i,0}$ stand for (e_{kmax+1}, b_i) and let $\mu_{i,j}$ stand for $(b_j^*, b_i)_{\rm mix}/(b_j^*, b_j^*)_{\rm mix}$ if $j > 0$. Note that initially $|\mu_{i,j}| \le 1$ for $i \le kmax$, $0 \le j \le kmax$. We will estimate $|\mu_{i,j}|$ as k descends.

Say $k > isodim + 1$ and modulo $\mathbb{R}v$ the vector b_k is linearly dependent on the b_i with $i < k$. Let us compute with b_k, b_{k-1} modulo $V = \mathbb{R}v + \sum_{i=1}^{k-2} \mathbb{R}b_i$. We have $b_k \equiv \mu_{k,k-1} b_{k-1}^*$ and $b_{k-1} \equiv b_{k-1}^*$ modulo V. With the extended euclidean algorithm of [C] we find an integer matrix $\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}$ of determinant one so that $\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \begin{pmatrix} 1 \\ \mu_{k,k-1} \end{pmatrix} = \begin{pmatrix} 0 \\ -1/r_k \end{pmatrix}$ where r_k is the index of $\mathbb{Z}$ in $\mathbb{Z} + \mathbb{Z}\mu_{k,k-1}$. More specifically, one has $\begin{pmatrix} \delta & -\beta \\ -\gamma & \alpha \end{pmatrix} \begin{pmatrix} 0 \\ -1/r_k \end{pmatrix} = \begin{pmatrix} 1 \\ \mu_{k,k-1} \end{pmatrix}$ so $\beta = r_k$ and $\alpha = -r_k\mu_{k,k-1}$. By [C] we have $|\gamma| \le |\mu_{k,k-1} r_k|$ and $|\delta| \le r_k$. (Here we assume for simplicity that $\mu_{k,k-1}$ is not zero.)

Now put $c_{k-1} = \alpha b_{k-1} + \beta b_k$ and $c_k = \gamma b_{k-1} + \delta b_k$. The algorithm will tell us to replace b_k with c_k and b_{k-1} with c_{k-1}. We want to estimate the resulting new $\mu_{i,j}$, which we call $\nu_{i,j}$. For i different from k, $k-1$ nothing changes. Further $|\nu_{k-1,j}| = |\alpha\mu_{k-1,j} + \beta\mu_{k,j}| \le r_k|\mu_{k,k-1}\mu_{k-1,j}| + r_k|\mu_{k,j}|$ and $|\nu_{k,j}| = |\gamma\mu_{k-1,j} + \delta\mu_{k,j}| \le r_k|\mu_{k,k-1}\mu_{k-1,j}| + r_k|\mu_{k,j}|$, which is the same bound.

Remark 7.1 During `trickledown` the d_i are repeatedly replaced by divisors. As we are recording the $\mu_{i,j}$ with $j > isodim$ as fractions with a denominator $d_{j-isodim}$, this means that one has to update the numerators too. By remembering the r_k one can postpone all this updating until the end of `trickledown`, processing the product of the corrections, rather than each correction separately.

Lemma 7.2 *As k descends we have*

1. $|\mu_{i,j}| \le 1$ *for* $k > i > j \ge 0$,
2. $|\mu_{k,j}| \le \sqrt{B}(\sqrt{n}(B+1))^{n/2} \prod_{i=k+1}^{kmax+1}(2r_i)$ *for* $k > j \ge 0$,
3. $|\mu_{i,j}| \le 2^n n^{n/4}(B+1)^n$ *for* $k \le i > j \ge 0$.

Proof

Initially we have $k = kmax + 1$ and we estimate $|\mu_{k,j}|^2 \leq B(b_j^*, b_j^*)_{\text{mix}}^{-1} \leq B(\sqrt{n}(B+1))^n$. Now assume the estimates are true for the present k. We get $|\nu_{k-1,j}| \leq r_k|\mu_{k,k-1}\mu_{k-1,j}| + r_k|\mu_{k,j}| \leq 2r_k \max_j |\mu_{k,j}|$ which takes care of $|\nu_{k-1,j}|$. As $\prod_{k=isodim+2}^{kmax+1} r_k^2$ is the ratio by which d_{rank} drops during `trickledown`, it is at most B^{rank}. So $|\nu_{k,j}| \leq \sqrt{B}(\sqrt{n}(B+1))^{n/2}2^n B^{rank/2}$. $\square$

Remark 7.3 Experiments show it is wise to insert a reduce step to make that $|\mu_{k,j}| \leq 1/2$ for $j \neq 0$.

7.4 Increasing *isodim*

When k has reached $isodim + 1$ it is time to increase $isodim$ by one and pass to a new $(\ ,\ )_{\text{mix}}$. But first use the estimates of the $\mu_{i,j}$ to estimate $(b_i, b_i)_{\text{mix}}$ and $(\mu_{i,0}e_{kmax+1}, \mu_{i,0}e_{kmax+1})_{\text{mix}}$, next $(b_i - \mu_{i,0}e_{kmax+1}, b_i - \mu_{i,0}e_{kmax+1})$ by means of Lemma 5.2, and finally (b_i, b_i).

Now change $isodim$, $kmax$, $(\ ,\ )_{\text{mix}}$. We have to compute the new $\mu_{j,isodim}$. They can be estimated, as we have an estimate for (b_j, b_j) and for $(b_{isodim}^*, b_{isodim}^*)_{\text{mix}}^{-1}$.

Finally we reduce to the case $|\mu_{i,j}| \leq 1/2$ for $i > j$. During this reduction the maximum of the $|\mu_{i,j}|$ gets at most 2^n as large by the argument in [LLL].

We have seen that all the integers that are encountered have bit length $\mathcal{O}(n\log(nB))$.

8 Implementation

For further details of implementation see the Mathematica code
`http://www.math.uu.nl/people/vdkallen/LLLsmall.m.gz`
or the GP/PARI code
`http://www.math.uu.nl/people/vdkallen/extendedlll.gp.gz`

References

[C] H. Cohen, *A course in computational Algebraic Number Theory*, Graduate Texts in Mathematics 138, Springer 1993.

[LLL] A.K. Lenstra, H.W. Lenstra Jr. and L. Lovász, *Factoring polynomials with rational coefficients*, Math. Ann. **261** (1982), 515–534.

[P] M. Pohst, *A modification of the LLL-algorithm*, J. Symb. Comp. **4** (1987), 123–128.