A Polynomial Time Version of LLL with Deep Insertions

Urs Wagner
University of Zurich, Applied Algebra Group

WCC 2013

Joint work with: Felix Fontein, UZH Applied Algebra Group
Michael Schneider, TU Darmstadt

Definitions

Definition

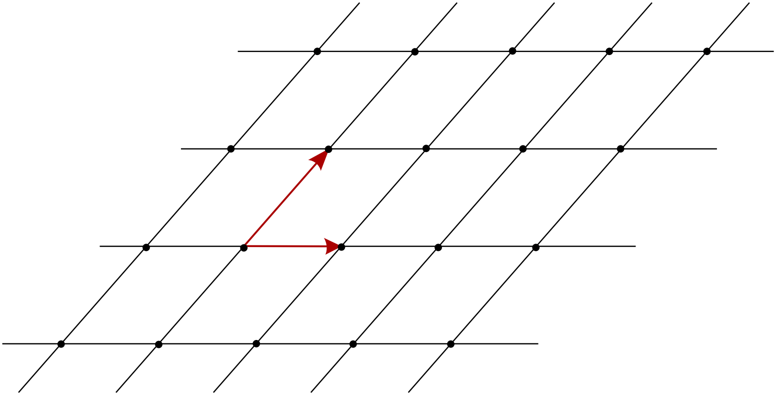
A **lattice** $\mathcal{L} \subset \mathbb{R}^n$ is given by the integer linear combinations of a set of linearly independent $b_1, \ldots, b_n \in \mathbb{R}^n$:

$$\mathcal{L}(b_1, \ldots, b_n) = \left\{ \sum_{i=1}^{d} x_i b_i : x_i \in \mathbb{Z} \right\}.$$

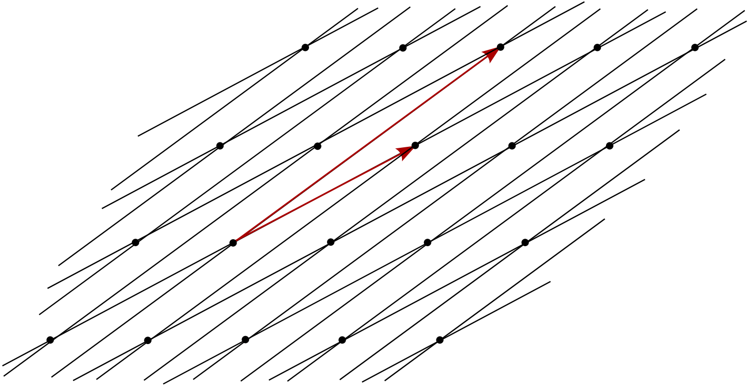
The vectors $b_1, \ldots, b_n$ are called **basis** of the lattice.

- Basis is not unique. Let $B = [b_1, \ldots, b_n]$ be a column matrix representing the basis of some lattice $\mathcal{L}$, then for all $U \in \mathrm{GL}_n(\mathbb{Z})$, BU represents another basis of $\mathcal{L}$.
- The **volume** of a lattice is invariant under the different bases: $\mathrm{vol}(\mathcal{L}) = \sqrt{\det(B^T B)}$.

Lattice



Lattice



Shortest Vector Problem (SVP)

Definition

The **first minimum** $\lambda_1(\mathcal{L})$ of a lattice $\mathcal{L}$ is defined as the length of the shortest vector in $\mathcal{L}$.

Definition

The **shortest vector problem (SVP)** asks for a nonzero lattice vector $v \in \mathcal{L}(B)$ such that $\|v\| = \lambda_1\big(\mathcal{L}(B)\big)$.

- NP-hard (under randomized reductions).
- Already determining λ_1 is hard.
- LLL algorithm to solve SVP approximately in polynomial time.

Shortest Vector Problem cont.

Definition

The **Hermite constant** γ_n is defined as the supremum of $\frac{\lambda_1(\mathcal{L})^2}{\mathrm{vol}(\mathcal{L})^{2/n}}$ over all rank-n lattices.

- I.e. $\lambda_1(\mathcal{L}) \leq \sqrt{\gamma_n}\mathrm{vol}(\mathcal{L})^{1/n}$.
- γ_n is known for $n = 2, 3, 4, 5, 6, 7, 8, 24$.
- Upper bound: $\gamma_n \leq 1 + \frac{n}{4}$.

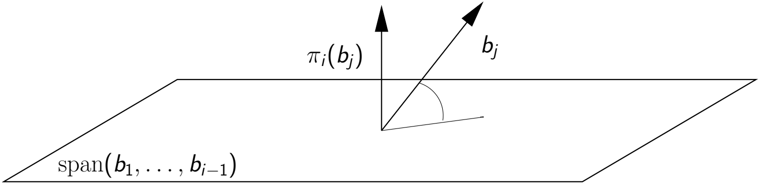
Definition

Given a lattice $\mathcal{L}$ and a factor $\alpha > 0$, the **Hermite-SVP** asks for a nonzero lattice vector $v \in \mathcal{L}$ such that $\|v\| \leq \alpha \cdot \mathrm{vol}(\mathcal{L})^{1/n}$.

- Known approximation algorithms (such as LLL and BKZ) achieve $\alpha = c^n$ for some $c > 1$.
- We call c the Hermite factor constant.

Orthogonal projection π_i

By $\pi_i : \mathbb{R}^n \longrightarrow \operatorname{span}(b_1, \ldots, b_{i-1})^\perp$, we denote the orthogonal projection as usual. In particular $b_i^* = \pi_i(b_i)$.



The potential of a lattice basis

Definition

The **potential** $\mathrm{Pot}(B)$ of a lattice basis $B = [b_1, \ldots, b_n]$ is defined as

$$\mathrm{Pot}(B) := \prod_{i=1}^{n} \mathrm{vol}\big(\mathcal{L}(b_1, \ldots, b_i)\big)^2$$

- For $1 \leq k < \ell \leq n$, adding an integer multiple of the k-th basis vector to the ℓ-th basis vector does not change the potential of the basis.
- I.e. Size reduction does not change the potential of the basis:

$$b_\ell \leftarrow b_\ell - \sum_{k=1}^{\ell-1} \lfloor \mu_{\ell,k} \rceil b_k$$

- Permutation of basis vectors does!

A class of permutations

For $1 \leq k \leq \ell \leq n$ we define a class of elements $\sigma_{k,\ell} \in S_n$ as follows:

$$\sigma_{k,\ell}(i) = \begin{cases} i & \text{for} \quad i < k \text{ or } i > \ell\,, \\ \ell & \text{for} \quad i = k\,, \\ i-1 & \text{for} \quad k < i \leq \ell\,. \end{cases}$$

Let $1 \leq k \leq \ell \leq n$ and $B = [b_1, \ldots, b_n]$, then

$$\begin{array}{rl} B & = [\; b_1 \;\ldots\; b_{k-1} \;\; b_k \;\; b_{k+1} \;\ldots\;\ldots\; b_{\ell-1} \;\; b_\ell \;\; b_{\ell+1} \;\ldots\; b_n \;] \\ \sigma_{k,\ell} B & = [\; b_1 \;\ldots\; b_{k-1} \;\; b_\ell \;\; b_k \;\; b_{k+1} \;\ldots\;\ldots\; b_{\ell-1} \;\; b_{\ell+1} \;\ldots\; b_n \;] \end{array}$$

Lemma

Let $B = [b_1, \ldots, b_n]$ be a lattice basis, $\delta \in (1/4, 1]$. Then for $1 \leq k \leq \ell \leq n$

$$\mathrm{Pot}(\sigma_{k,\ell} B) = \mathrm{Pot}(B) \cdot \prod_{i=k}^{\ell} \frac{\|\pi_i(b_\ell)\|^2}{\|\pi_i(b_i)\|^2}.$$

Approximation algorithms

Definition

A basis $B = [b_1, \dots, b_n]$ whose Gram-Schmidt coefficients $\mu_{ij} = \frac{\langle \pi_j(b_j), b_i \rangle}{\|\pi_j(b_j)\|^2}$, $1 \le j < i \le n$ satisfy

$$|\mu_{ij}| \le 1/2,$$

is called

- δ-**LLL reduced** if for $1 \le k < n$:

$$\delta \cdot \|\pi_k(b_k)\|^2 \le \|\pi_k(b_{k+1})\|^2 \qquad \big(\Leftrightarrow \delta \cdot \mathrm{Pot}(B) \le \mathrm{Pot}(\sigma_{k,k+1} B)\big).$$

- δ-**PotLLL reduced** if $1 \le k < \ell \le n$:

$$\delta \cdot \mathrm{Pot}(B) \le \mathrm{Pot}(\sigma_{k,\ell} B).$$

- δ-**DeepLLL**-β **reduced** if $1 \le k < \ell \le n$ with $k \le \beta \wedge \ell - k \le \beta$:

$$\delta \cdot \|\pi_k(b_k)\|^2 \le \|\pi_k(b_\ell)\|^2.$$

- δ-**BKZ**-β **reduced** if $1 \le k \le n$:

$$\delta \cdot \|\pi_k(b_k)\|^2 \le \lambda_1\Big(\mathcal{L}\big(\pi_k(b_k), \dots, \pi_k(b_{\min(k+\beta-1,n)})\big)\Big).$$

LLL vs PotLLL

Algorithm 1: LLL

Input: Basis B, $\delta \in (1/4, 1]$
Output: A δ-LLL reduced basis.

$\ell \leftarrow 2$
while $\ell \leq n$ **do**
  Size-reduce(B)
  $k \leftarrow \ell - 1$
  if $\delta \cdot \mathrm{Pot}(B) > \mathrm{Pot}(\sigma_{k,\ell} B)$ **then**
    $B \leftarrow \sigma_{k,\ell} B$
    $\ell \leftarrow k$
  else
    $\ell \leftarrow \ell + 1$
  end
end
return B

Algorithm 2: PotLLL

Input: Basis B, $\delta \in (1/4, 1]$
Output: A δ-PotLLL reduced basis.

$\ell \leftarrow 2$
while $\ell \leq n$ **do**
  Size-reduce(B)
  $k \leftarrow \mathrm{argmin}_{1 \leq j \leq \ell} \mathrm{Pot}(\sigma_{j,\ell} B)$
  if $\delta \cdot \mathrm{Pot}(B) > \mathrm{Pot}(\sigma_{k,\ell} B)$ **then**
    $B \leftarrow \sigma_{k,\ell} B$
    $\ell \leftarrow k$
  else
    $\ell \leftarrow \ell + 1$
  end
end
return B

- One might think of different ways to compute a PotLLL reduced basis.
- Future work: $k \leftarrow \min \left\{ k : \delta \cdot \mathrm{Pot}(B) > \mathrm{Pot}(\sigma_{k,\ell} B) \right\}$

Worst-case behaviour

Running time:

- LLL and PotLLL have polynomial running time for $\delta < 1$.
- No useful upper bound known for BKZ and DeepLLL.

SVP approximation factor (case $\delta = 1$):

LLL: $\|b_1\| \leq \left(\sqrt{\gamma_2}\right)^{n-1} \mathrm{vol}\big(\mathcal{L}(B)\big)^{1/n} = \left(\sqrt{\frac{4}{3}}\right)^{n-1} \mathrm{vol}\big(\mathcal{L}(B)\big)^{1/n}$

PotLLL: $\|b_1\| \leq \left(\sqrt{\gamma_2}\right)^{n-1} \mathrm{vol}\big(\mathcal{L}(B)\big)^{1/n}$

DeepLLL: $\|b_1\| \leq \left(\sqrt{\gamma_2}\right)^{n-1} \mathrm{vol}\big(\mathcal{L}(B)\big)^{1/n}$

BKZ-β: $\|b_1\| \leq \left(\sqrt{\gamma_\beta}\right)^{(n-1)/(\beta-1)+1} \mathrm{vol}\big(\mathcal{L}(B)\big)^{1/n}$.

- Critical bases exist for LLL, DeepLLL and PotLLL!

Hermite factor constant

- N. Gama, P. Nguyen: Predicting Lattice Reduction (Eurocrypt 2008):
- Practical behaviour much better.
- Practical Hermite factor still exponential in n, i.e. $\|b_1\| = c^n \cdot \mathrm{vol}\big(\mathcal{L}(B)\big)^{1/n}$, where c depends on reduction algorithm.

	upper bound	empirical
LLL	1.0754	1.0219
BKZ-20	1.0337	1.0128
DeepLLL-50	1.0754	1.011

- Using V. Shoups NTL library.

PotLLL vs the rest

- Our own independent implementation.
- Dimensions 40, 50,. . . , 400.
- 50 random lattices in each dimension (challenge lattices[1] with seed= $1, \ldots, 50$).
- Reduction algorithms: PotLLL, LLL, DeepLLL-β, BKZ-β.
- Hermite factor constant $\|b_1\| = c^n \cdot \mathrm{vol}\big(\mathcal{L}(B)\big)^{1/n}$

Dimension	$n = 100$	$n = 200$	$n = 300$	$n = 400$
LLL	1.0187	1.0204	1.0212	1.0212
BKZ-5	1.0154	1.0160	1.0163	—
PotLLL	1.0146	1.0151	1.0153	1.0154
DeepLLL-5	1.0138	1.0146	1.0150	—
BKZ-10	1.0140	1.0144	1.0145	—
DeepLLL-10	1.0128	1.0135	—	—

[1] http://www.latticechallenge.org/svp-challenge

Outline

Lattices and the shortest vector problem

Potential of a lattice basis

PotLLL

Practical behaviour

Implementation and conclusion

LLL vs PotLLL vs DeepLLL

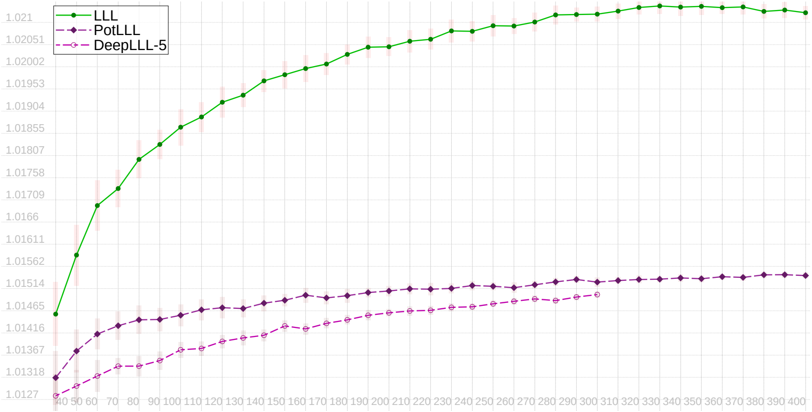


Figure: Hermite factor constant

LLL vs PotLLL vs DeepLLL

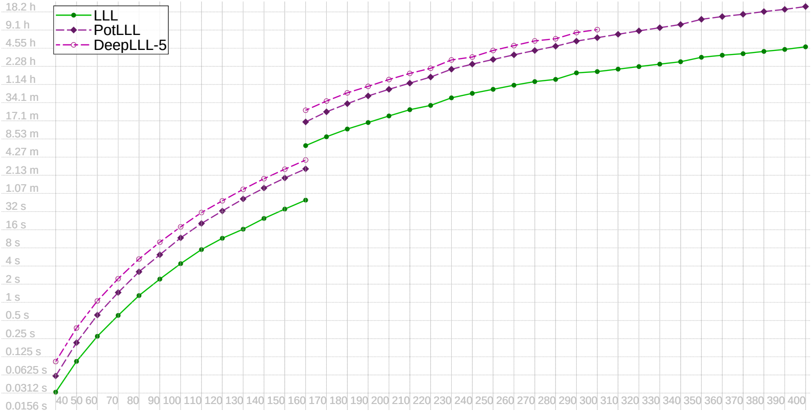


Figure: Time

PotLLL vs BKZ

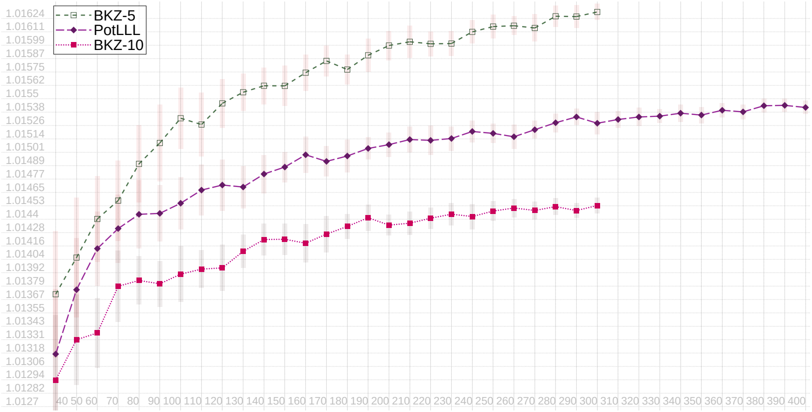


Figure: Hermite factor constant

PotLLL vs BKZ

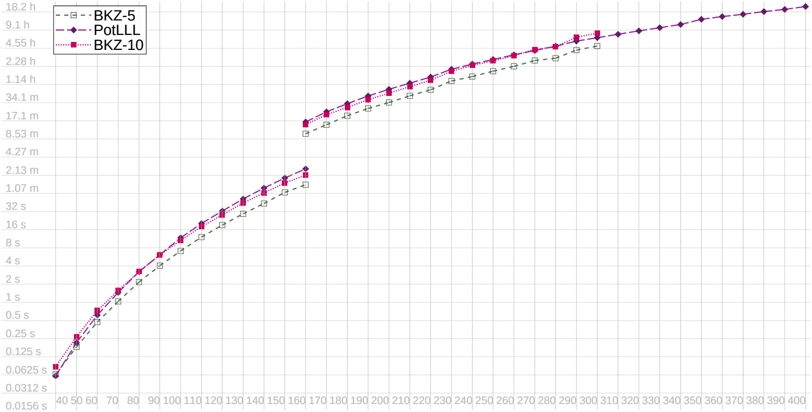
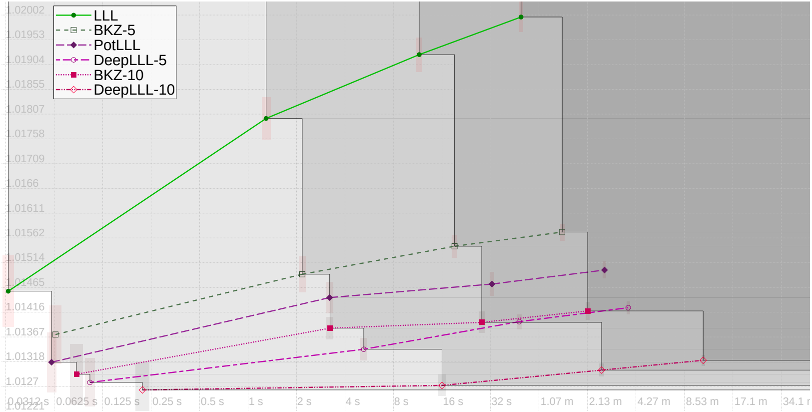
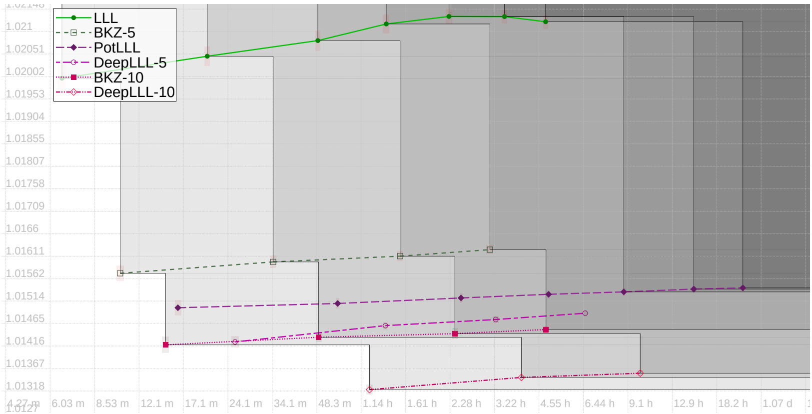


Figure: Time

Overview


LLL
BKZ-5
PotLLL
DeepLLL-5
BKZ-10
DeepLLL-10
1.02002
1.01953
1.01904
1.01855
1.01807
1.01758
1.01709
1.0166
1.01611
1.01562
1.01514
1.01465
1.01416
1.01367
1.01318
1.0127
1.01221
0.0312 s
0.0625 s
0.125 s
0.25 s
0.5 s
1 s
2 s
4 s
8 s
16 s
32 s
1.07 m
2.13 m
4.27 m
8.53 m
17.1 m
34.1 m

Overview


LLL
BKZ-5
PotLLL
DeepLLL-5
BKZ-10
DeepLLL-10
1.021
1.02051
1.02002
1.01953
1.01904
1.01855
1.01807
1.01758
1.01709
1.0166
1.01611
1.01562
1.01514
1.01465
1.01416
1.01367
1.01318
1.0127
4.27 m
6.03 m
8.53 m
12.1 m
17.1 m
24.1 m
34.1 m
48.3 m
1.14 h
1.61 h
2.28 h
3.22 h
4.55 h
6.44 h
9.1 h
12.9 h
18.2 h
1.07 d

Conclusion, further remarks

- First polynomial time version of LLL with deep insertions
- Step towards complexity analysis of DeepLLL.
- Extended experiments on practical behaviour of lattice reduction algorithms.
- Our implementation will be made public soon. On `http://user.math.uzh.ch/fontein/fplll-potlll/` corresponding extension of fplll is provided already.
- **Future work:** Different classes of permutations.

Implementation

- All experiments were run on Intel® Xeon® X7550 CPUs at 2 GHz on a shared memory machine.
- For dimensions 40 up to 160, we used `long double` arithmetic, and for dimensions 160 up to 400, we used MPFR.
- In dimension 160, we did the experiments both using `long double` and MPFR arithmetic. The reduced lattices did not differ.
- In dimension 170, floating point errors prevented the `long double` arithmetic variant to complete on some of the lattices.

Thanks!