What is Lattice Point Enumeration up to These Days?

. . . being a subroutine of BKZ

Martin R. Albrecht[a]

15 June 2021, Lattice Reunion

[a]based on joint work with Shi Bai, Pierre-Alain Fouque, Paul Kirchner, Jianwei Li, Joe Rowell, Damien Stehlé and Weiqiang Wen

Outline

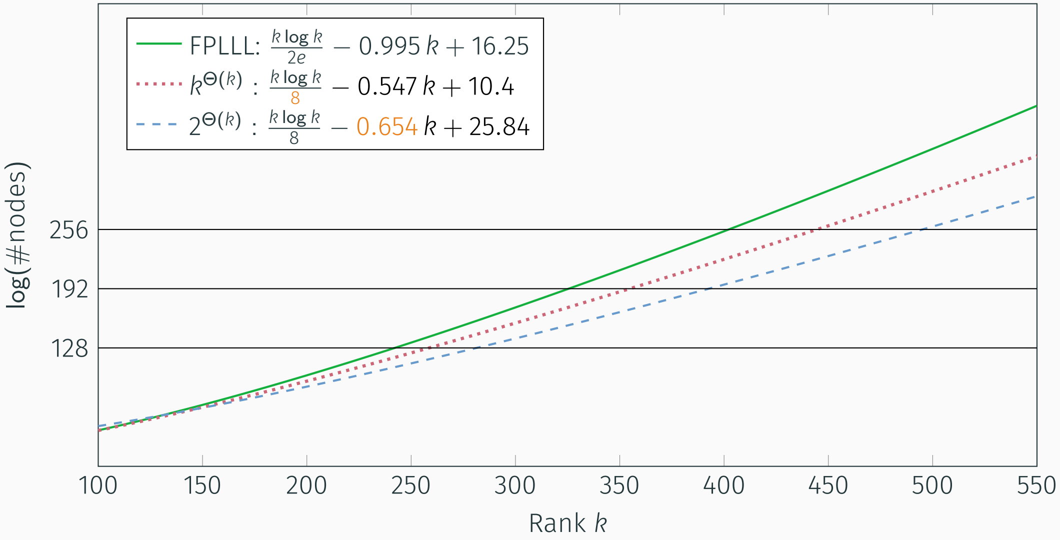
Preliminaries

Enumeration Recap

Super-exponential: $(1 + c) \cdot k$

Exponential: $(\alpha \cdot \mathrm{GH}(k_\alpha))^{\frac{1}{k_\alpha - 1}} \leq \mathrm{GH}(k)^{\frac{1}{k-1}}$

Headline



FPLLL: $\frac{k \log k}{2e} - 0.995\,k + 16.25$
$k^{\Theta(k)}$: $\frac{k \log k}{8} - 0.547\,k + 10.4$
$2^{\Theta(k)}$: $\frac{k \log k}{8} - 0.654\,k + 25.84$
log(#nodes)
256
192
128
100
150
200
250
300
350
400
450
500
550
Rank k

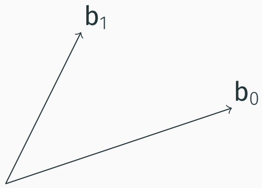
Preliminaries

Length of Gram-Schmidt Vectors

It will be useful to consider the lengths of the Gram-Schmidt vectors.

The vector $\mathbf{b}_i^*$ is the orthogonal projection of $\mathbf{b}_i$ to the space spanned by the vectors $\mathbf{b}_0, \ldots, \mathbf{b}_{i-1}$.

Informally, this means taking out the contributions in the directions of previous vectors $\mathbf{b}_0, \ldots, \mathbf{b}_{i-1}$.

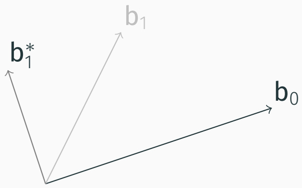


Length of Gram-Schmidt Vectors

It will be useful to consider the lengths of the Gram-Schmidt vectors.

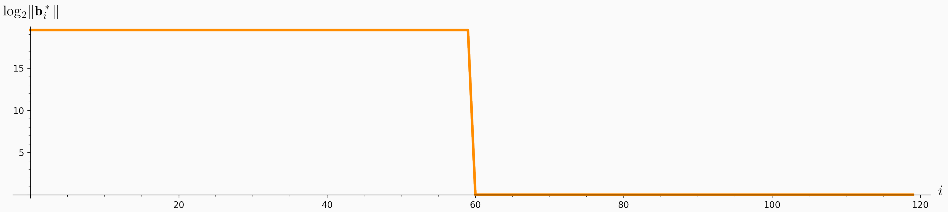
The vector $\mathbf{b}_i^*$ is the orthogonal projection of $\mathbf{b}_i$ to the space spanned by the vectors $\mathbf{b}_0, \ldots, \mathbf{b}_{i-1}$.

Informally, this means taking out the contributions in the directions of previous vectors $\mathbf{b}_0, \ldots, \mathbf{b}_{i-1}$.



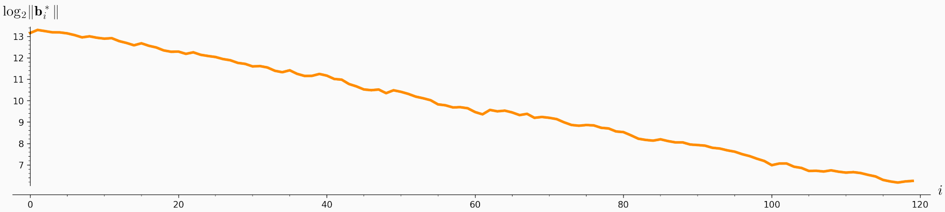
Example

```
sage: A = IntegerMatrix.random(120, "qary", k=60, bits=20)[::-1]
sage: M = GSO.Mat(A); M.update_gso()
sage: line([(i,log(r_, 2)/2) for i, r_ in enumerate(M.r())], **plot_kwds)
```



Example - LLL

```
sage: A = LLL.reduction(A)
sage: M = GSO.Mat(A); M.update_gso()
sage: line([(i,log(r_, 2)/2) for i, r_ in enumerate(M.r())], **plot_kwds)
```



BKZ Algorithm

$$\begin{pmatrix} \overbrace{\mathbf{b}_0 \quad \mathbf{b}_1 \quad \mathbf{b}_2 \quad \mathbf{b}_3 \quad \mathbf{b}_4}^{k=5} & \mathbf{b}_5 & \mathbf{b}_6 & \mathbf{b}_7 & \ldots \end{pmatrix}$$

Picture credit: Eamonn Postlethwaite

BKZ Algorithm

$$\left(\overbrace{\mathbf{b}_0 \quad \mathbf{b}_1 \quad \mathbf{b}_2 \quad \mathbf{b}_3 \quad \mathbf{b}_4}^{k = 5} \quad \mathbf{b}_5 \quad \mathbf{b}_6 \quad \mathbf{b}_7 \quad \dots \right)$$

Picture credit: Eamonn Postlethwaite

BKZ Algorithm

$$\begin{pmatrix} \underbrace{{\color{red}b_0} \quad b_1 \quad b_2 \quad b_3 \quad b_4}_{} & b_5 & b_6 & b_7 & \ldots \end{pmatrix}$$

$$k = 5$$

Picture credit: Eamonn Postlethwaite

BKZ Algorithm

$$\left(\begin{array}{ccccccccc} \color{red}{\mathbf{b}_0} & \underbrace{\mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 & \mathbf{b}_4 & \mathbf{b}_5}_{k=5} & \mathbf{b}_6 & \mathbf{b}_7 & \ldots \end{array} \right)$$

Picture credit: Eamonn Postlethwaite

BKZ Algorithm

$$\left(\color{red}{b_0} \;\; \overbrace{\underbrace{b_1 \;\; b_2 \;\; b_3 \;\; b_4 \;\; b_5}}^{k=5} \;\; b_6 \;\; b_7 \;\; \ldots \right)$$

Picture credit: Eamonn Postlethwaite

BKZ Algorithm

$$\left(\mathbf{b}_0 \quad \overbrace{\mathbf{b}_1 \quad \mathbf{b}_2 \quad \mathbf{b}_3 \quad \mathbf{b}_4 \quad \mathbf{b}_5}^{k=5} \quad \mathbf{b}_6 \quad \mathbf{b}_7 \quad \dots \right)$$

Picture credit: Eamonn Postlethwaite

BKZ Algorithm

$$\begin{pmatrix} \color{red}{b_0} & \color{red}{b_1} & \overbrace{b_2 \quad b_3 \quad b_4 \quad b_5 \quad b_6}^{k=5} & b_7 & \ldots \end{pmatrix}$$

Picture credit: Eamonn Postlethwaite

BKZ Algorithm

$$\left(\; b_0 \quad b_1 \quad \overbrace{b_2 \quad b_3 \quad b_4 \quad b_5 \quad b_6}^{k = 5} \quad b_7 \quad \ldots \;\right)$$

Picture credit: Eamonn Postlethwaite

BKZ Algorithm

$$\begin{pmatrix} \mathbf{b}_0 & \mathbf{b}_1 & \overbrace{\mathbf{b}_2 \quad \mathbf{b}_3 \quad \mathbf{b}_4 \quad \mathbf{b}_5 \quad \mathbf{b}_6}^{k=5} & \mathbf{b}_7 & \dots \end{pmatrix}$$

Picture credit: Eamonn Postlethwaite

BKZ Algorithm

```
def bkz_tour(B, k, e):
    for i in range(0, e):
        preprocess(B[i:i+k])
        v = svp(B[i:i+k])
        insert(v, B, i)

def bkz(B, k):
    while True:
        bkz_tour(B, k, d-1)
        if nothing_changed(B):
            break
```


- Claus-Peter Schnorr. A Hierarchy of Polynomial Time Lattice Basis Reduction Algorithms. In: *Theor. Comput. Sci.* 53 (1987), pp. 201–224
- Claus-Peter Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. In: *Math. Program.* 66 (1994), pp. 181–199. DOI: `10.1007/BF01581144`. URL: `https://doi.org/10.1007/BF01581144`
- Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. Analyzing Blockwise Lattice Algorithms Using Dynamical Systems. In: *CRYPTO 2011*. Ed. by Phillip Rogaway. Vol. 6841. LNCS. Springer, Heidelberg, Aug. 2011, pp. 447–464. DOI: `10.1007/978-3-642-22792-9_25`
- Jianwei Li and Phong Q. Nguyen. A Complete Analysis of the BKZ Lattice Reduction Algorithm. Cryptology ePrint Archive, Report 2020/1237. `https://eprint.iacr.org/2020/1237`. 2020

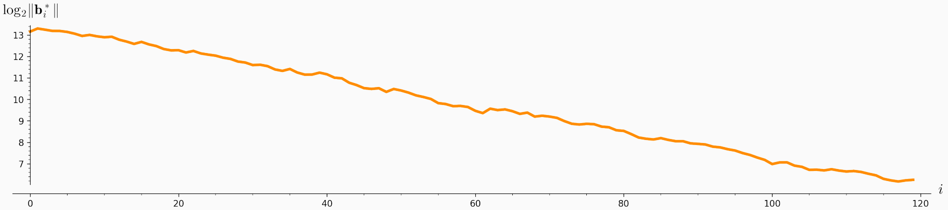
SD-BKZ Algorithm

```
def dual_bkz_tour(B, k, e):
    D = dual(B) # not actually needed
    for i in range(0, e):
        preprocess(D[i:i+k])
        v = svp(D[i:i+k])
        insert(v, D, i)
    B = dual(D)

 def sd_bkz(B, k):
    while True:
        bkz_tour(B, k, d-k)
        dual_bkz_tour(B, k, d-k)
        if nothing_changed(B):
            break
```


- Daniele Micciancio and Michael Walter. Practical, Predictable Lattice Basis Reduction. In: *EUROCRYPT 2016, Part I*. ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9665. LNCS. Springer, Heidelberg, May 2016, pp. 820–849. DOI: 10.1007/978-3-662-49890-3_31

GSA



Geometric Series Assumption: The shape after lattice reduction is a line with a flatter slope as lattice reduction gets stronger.[1]

[1] Claus-Peter Schnorr. Lattice Reduction by Random Sampling and Birthday Methods. In: *STACS 2003, 20th Annual Symposium on Theoretical Aspects of Computer Science, Berlin, Germany, February 27 - March 1, 2003, Proceedings*. Ed. by Helmut Alt and Michel Habib. Vol. 2607. Lecture Notes in Computer Science. Springer, 2003, pp. 145–156. DOI: `10.1007/3-540-36494-3_14`. URL: `http://dx.doi.org/10.1007/3-540-36494-3_14`.

Heuristics

Heuristic 1 The GSA holds for the first $n - k$ vectors after SD-BKZ-k reduction.

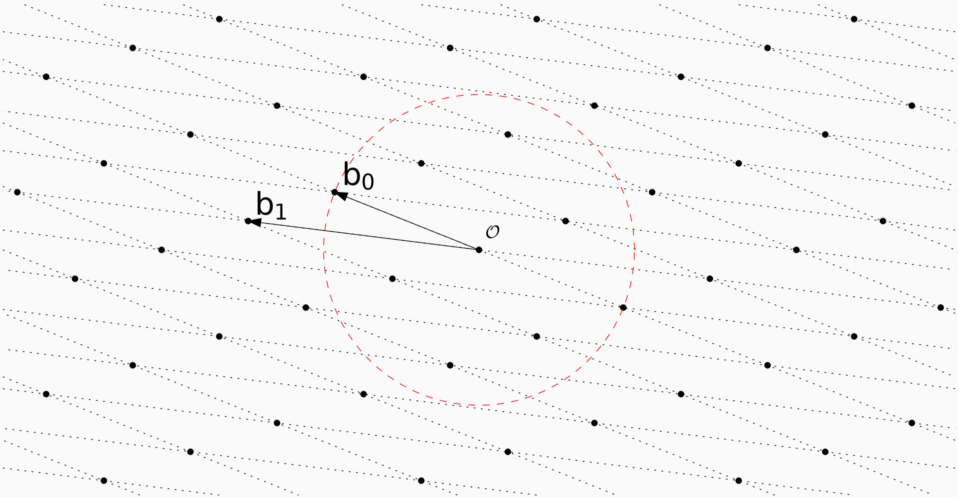
Heuristic 2 The GSA holds.

Scope

Heuristics only used in theorems, our simulations make no heuristic assumptions. Our simulations are backed up by experimental evidence (in small-ish dimensions) from implementations.

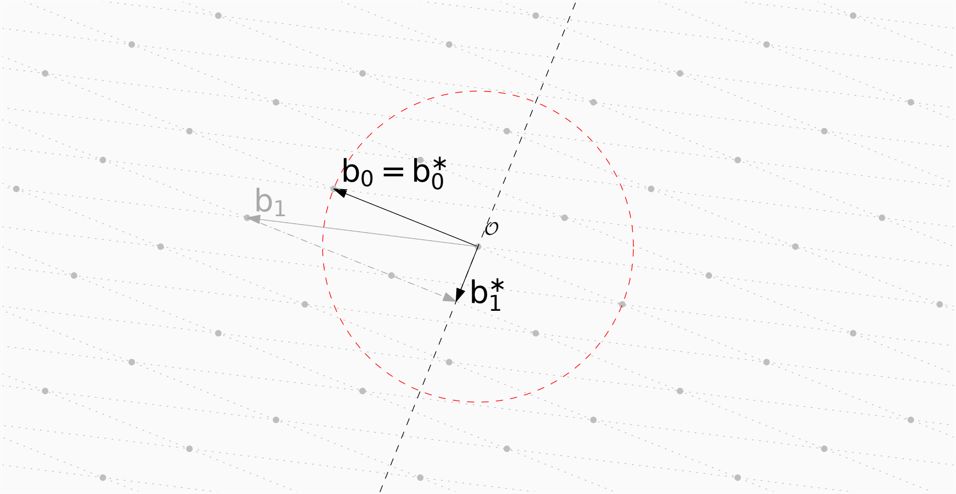
Enumeration Recap

Enumeration I – Pick a radius



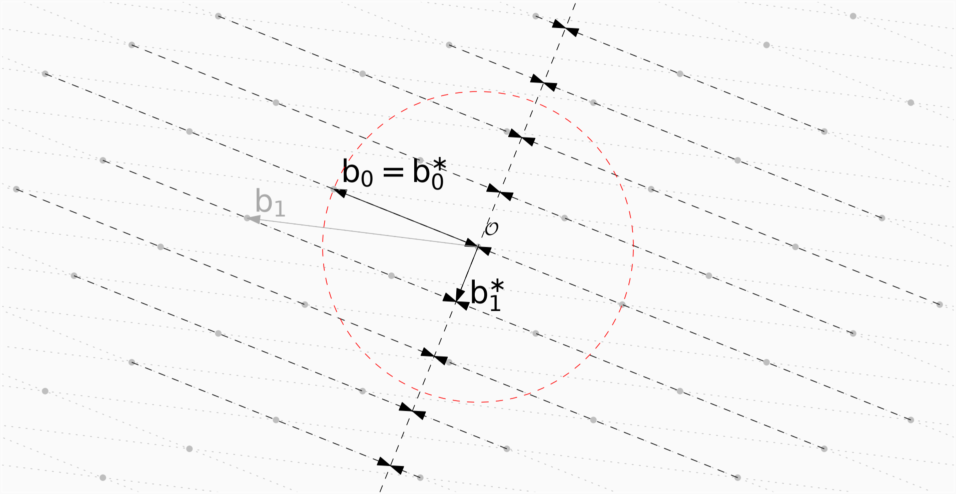
Picture credit: Joop van de Pol

Enumeration II – Project basis



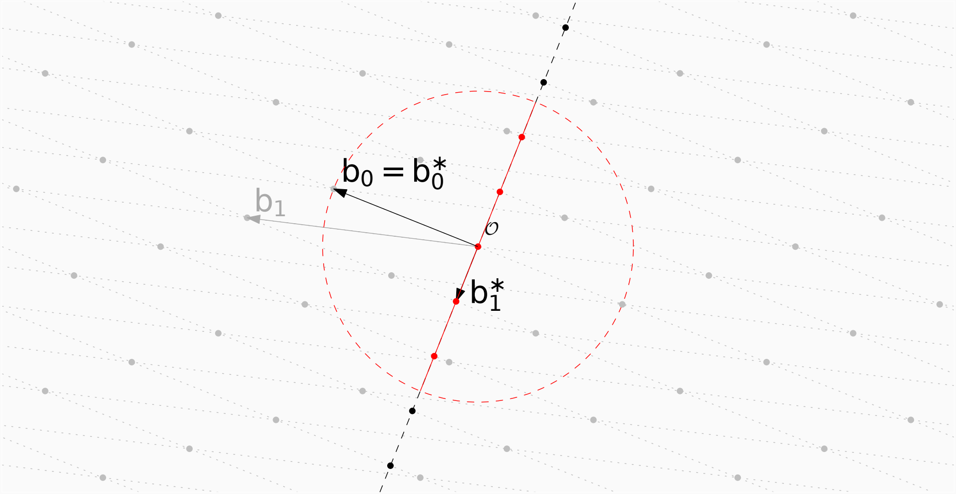
Picture credit: Joop van de Pol

Enumeration III – Project lattice



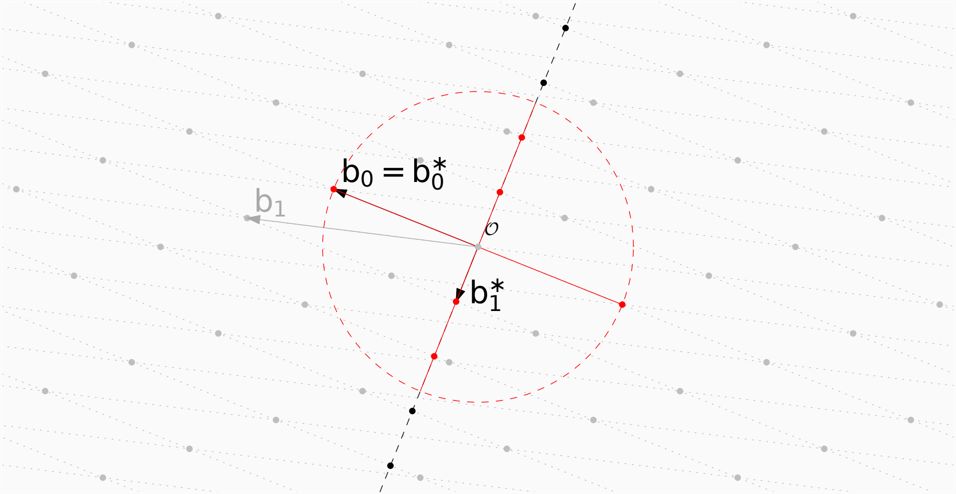
Picture credit: Joop van de Pol

Enumeration IV – Enumerate projections



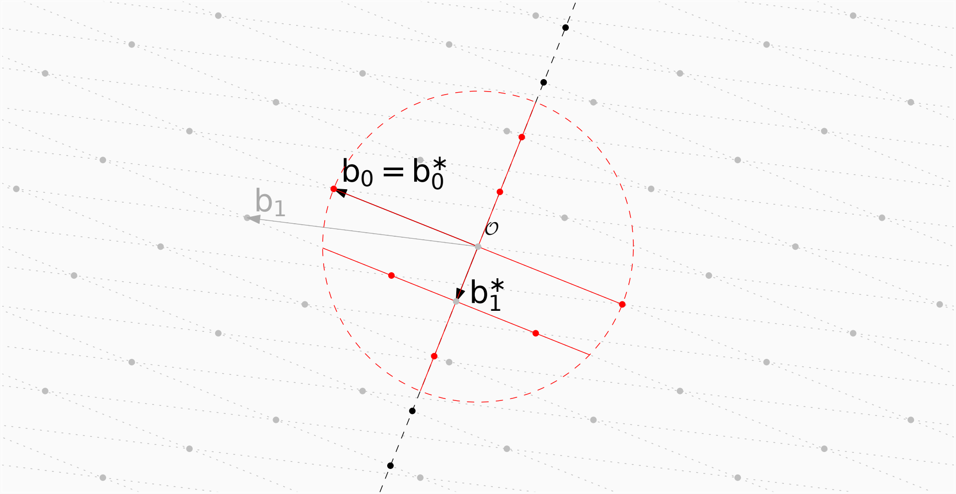
Picture credit: Joop van de Pol

Enumeration V – For each lift and enumerate



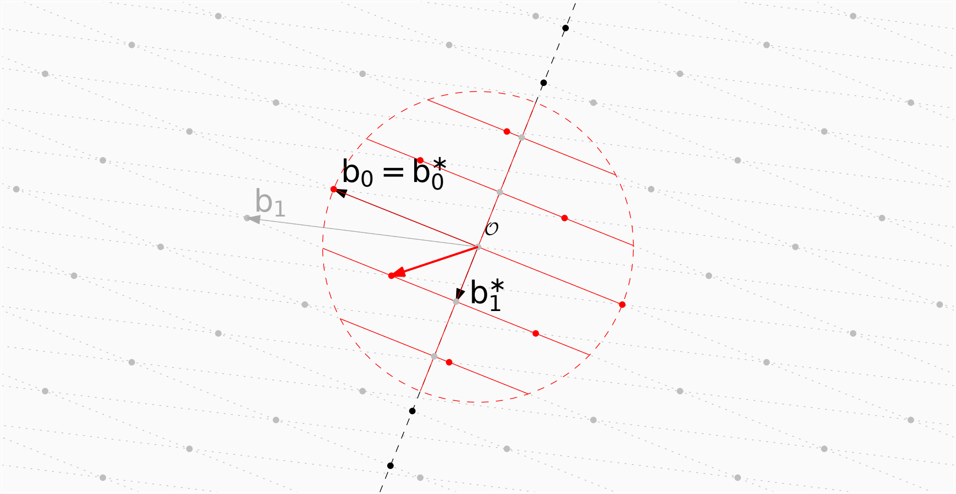
Picture credit: Joop van de Pol

Enumeration V – For each lift and enumerate



Picture credit: Joop van de Pol

Enumeration VI – Keep shortest



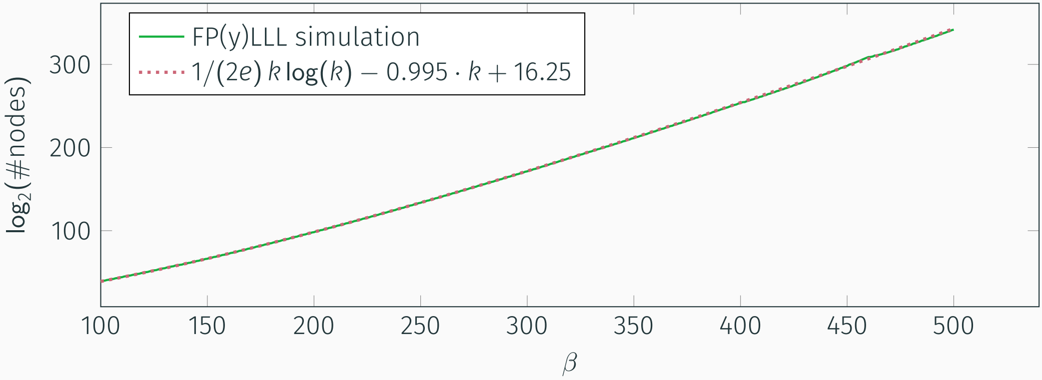
Picture credit: Joop van de Pol

Enumeration Cost: $k^{k/(2e)+o(k)}$ I

"We obtain a new worst-case complexity upper bound, as well as the first worst-case complexity lower bound, both of the order d of $2^{O(d)} \cdot d^{\frac{d}{2e}}$ (up to polynomial factors) bit operations, where d is the rank of the lattice."[2]

[2] Full version of Guillaume Hanrot and Damien Stehlé. Improved Analysis of Kannan's Shortest Lattice Vector Algorithm. In: *CRYPTO 2007*. Ed. by Alfred Menezes. Vol. 4622. LNCS. Springer, Heidelberg, Aug. 2007, pp. 170–186. DOI: `10.1007/978-3-540-74143-5_10`, available at `http://perso.ens-lyon.fr/damien.stehle/KANNAN_EXTENDED.html`

Enumeration Cost: $k^{k/(2e)+o(k)}$ II



Martin R. Albrecht, Shi Bai, Pierre-Alain Fouque, Paul Kirchner, Damien Stehlé, and Weiqiang Wen. Faster Enumeration-Based Lattice Reduction: Root Hermite Factor $k^{1/(2k)}$ Time $k^{k/8+o(k)}$. In: *CRYPTO 2020, Part II*. ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12171. LNCS. Springer, Heidelberg, Aug. 2020, pp. 186–212. DOI: 10.1007/978-3-030-56880-1_7

Super-exponential: $(1 + c) \cdot k$

Paper

Martin R. Albrecht, Shi Bai, Pierre-Alain Fouque, Paul Kirchner, Damien Stehlé, and Weiqiang Wen. Faster Enumeration-Based Lattice Reduction: Root Hermite Factor $k^{1/(2k)}$ Time $k^{k/8+o(k)}$. In: *CRYPTO 2020, Part II*. ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12171. LNCS. Springer, Heidelberg, Aug. 2020, pp. 186–212. DOI: 10.1007/978-3-030-56880-1_7

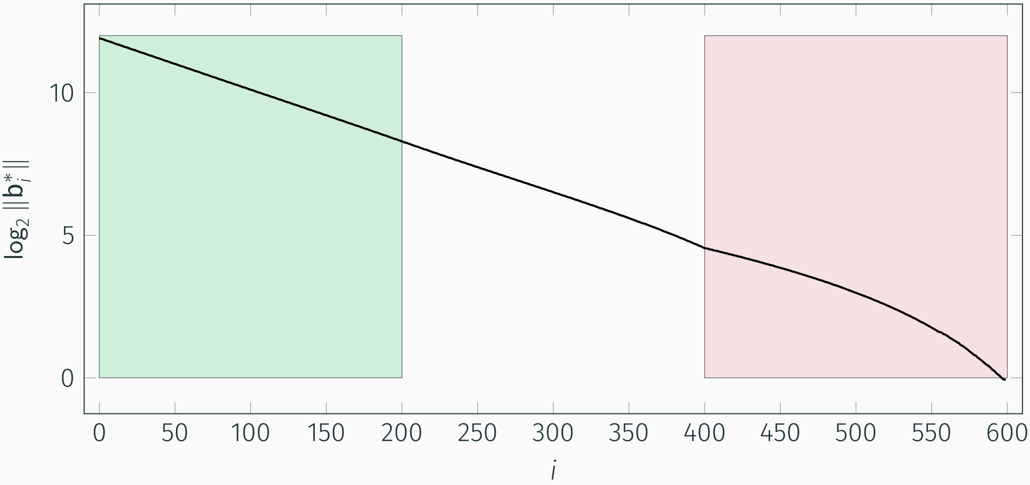
Enumeration Heuristic Best-Case Complexity

"Some authors favor the hypothesis that the average behaviour of an HKZ-reduced basis is rather a geometric decrease of the $\|\mathbf{b}_i^*\|$'s, i.e., roughly $\|\mathbf{b}_i^*\| \approx d^{\frac{i}{d}} \cdot \|\mathbf{b}_1\|$. With such a basis, solving SVP by Kannan's algorithm would have a $2^{O(d)} \cdot d^{\frac{d}{8}}$ complexity."[2]

One Interpretation

$\approx$ our result immediately holds if you simply assume the GSA.

$1/8 = 0.125$ v $1/(2e) \approx 0.184$

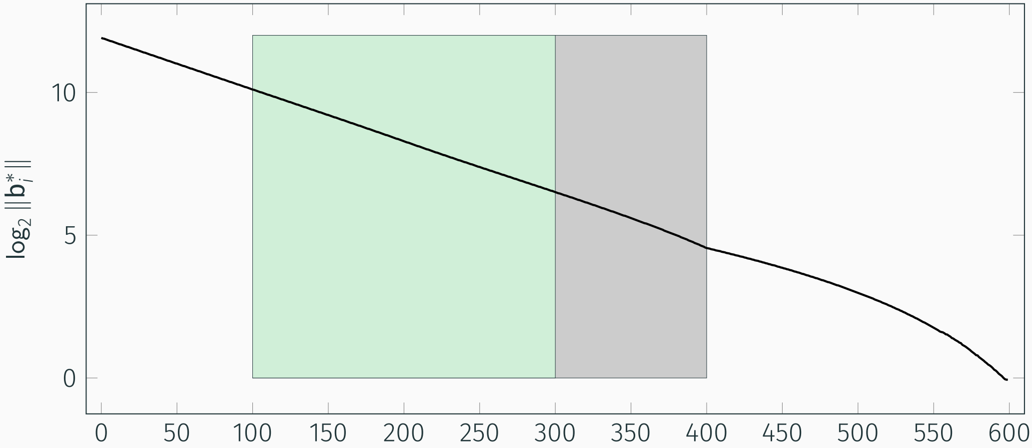


Why we can't have Nice Things

1. We run enumeration many times each succeeding with low probability of success and re-randomise in between: this destroys the nice GSA-line shape
   - Thus, before enumerating a local block, we run some local preprocessing with some block size $k' < k$
2. In the sandpile model,[3] as the algorithm proceeds through the indices i, a "bump" accumulates from index $i + 1$ onward.

[3] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. Analyzing Blockwise Lattice Algorithms Using Dynamical Systems. In: *CRYPTO 2011*. Ed. by Phillip Rogaway. Vol. 6841. LNCS. Springer, Heidelberg, Aug. 2011, pp. 447–464. DOI: 10.1007/978-3-642-22792-9_25.

Idea: Overshoot Preprocessing



Preprocessing in dimension $(1 + c) \cdot k$ for enumeration in dimension k.

Theorem

Theorem (Informal)

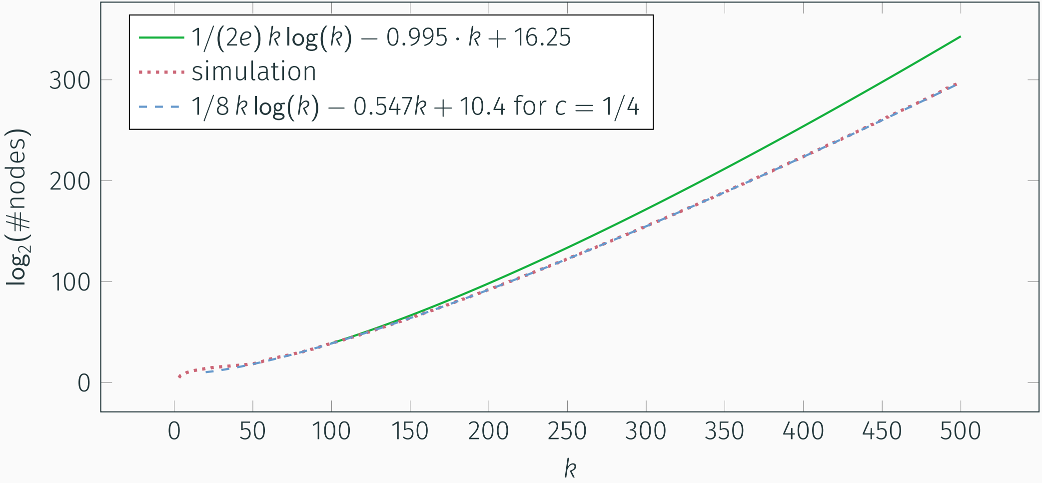
Under Heuristic 1 there exists an algorithm that achieves root Hermite factor $k^{\frac{1}{2k}}$ in time $k^{k/8+o(k)}$.

- Heuristic 1: The GSA holds for the first $n-k$ vectors after SD-BKZ-k reduction.
- Approach:
  - Define $k_0 = x_0 \cdot k$ with $x_0 = \frac{e}{4}(1+o(1))$ and, for all $i \geq 1$:

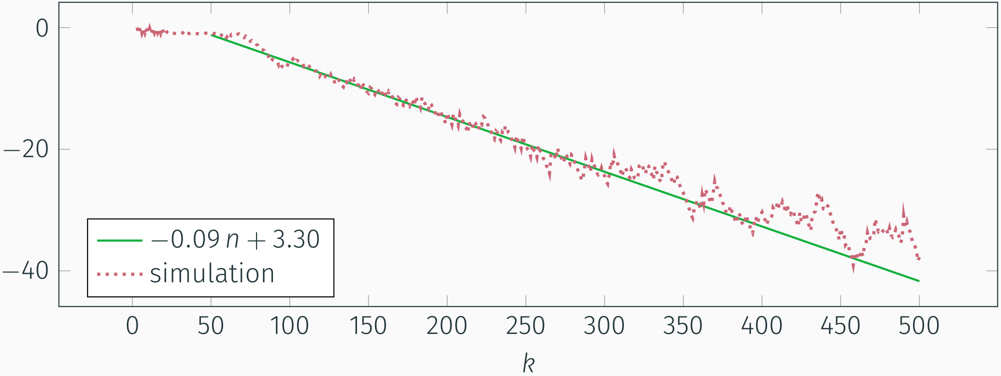
$$k_i = \lceil x_i \cdot k \rceil \quad \text{with} \quad x_i = x_{i-1} + \sqrt{\frac{x_{i-1}}{i}}.$$

  - so we start with $k_0^{k_0/(2e)} \approx k^{k/8}$
  - preprocess with increasing block sizes k_i
  - enumerate over the "line" part of the shape only

Practical Performance (Simulation)

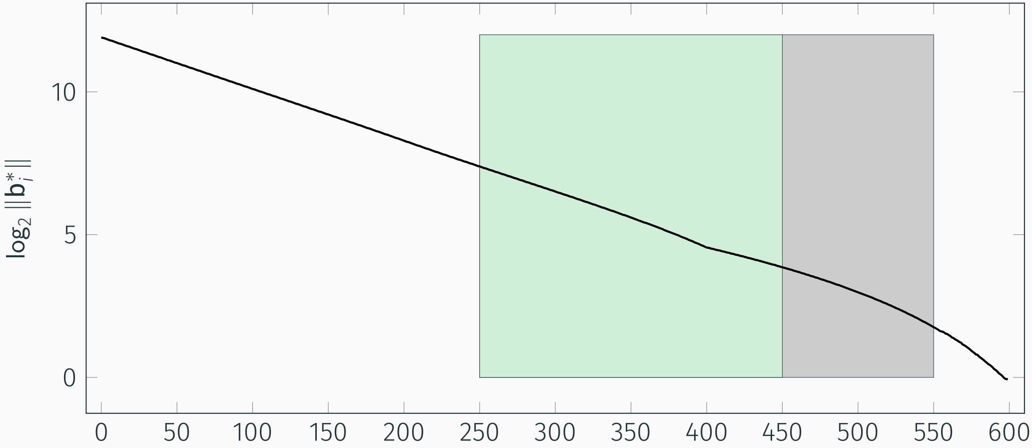


Not-so-extreme Pruning

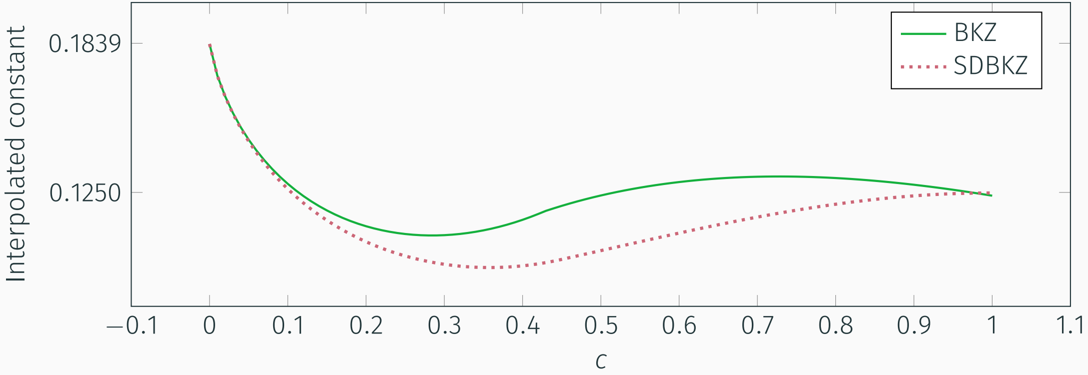


Success probability of a single enumeration in log scale.

Can we do better?



Can we do better?



Leading constant assuming **free preprocessing**.

Exponential:

$$\left(\alpha \cdot \mathrm{GH}(k_\alpha)\right)^{\frac{1}{k\alpha - 1}} \leq \mathrm{GH}(k)^{\frac{1}{k-1}}$$

Paper

Martin R. Albrecht, Shi Bai, Jianwei Li, and Joe Rowell. Lattice Reduction with Approximate Enumeration Oracles: Practical Algorithms and Concrete Performance. Cryptology ePrint Archive, Report 2020/1260. https://eprint.iacr.org/2020/1260. 2020, to appear at CRYPTO'21.

$\alpha \cdot GH(k)$ v $GH(k)$

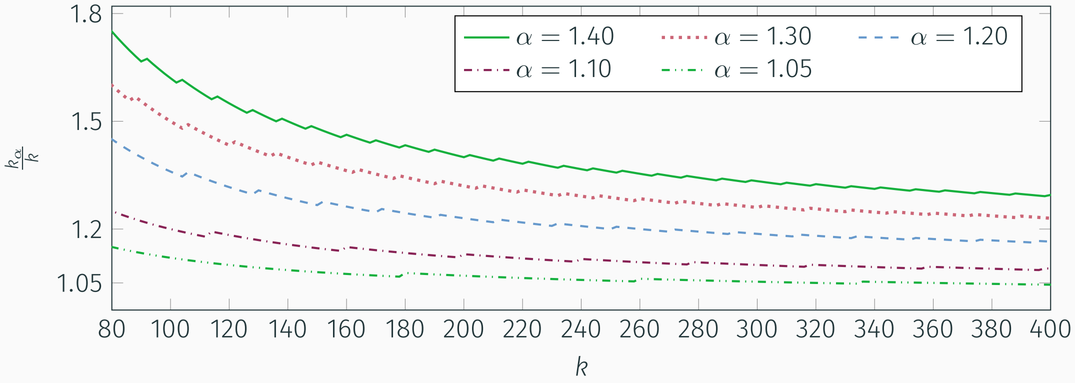
Corollary

Let Λ be a full-rank lattice in $\mathbb{R}^n$. Let $\alpha \geq 1$ and $\rho \in (0, \frac{1}{2})$ such that $4\alpha^4\rho(1-\rho) < 1$. Let $R = \mathrm{GH}(\Lambda)$, $R_\alpha = \alpha \cdot \mathrm{GH}(\Lambda)$ and $f(i)$ be a certain pruning function.

Under Heuristic 2, the time complexity of enumeration with radius R_α is less than that with radius R by a multiplicative factor $\alpha^{n/2}$ (up to some polynomial factor).

Corollary of Theorem 6 in Jianwei Li and Phong Q. Nguyen. A Complete Analysis of the BKZ Lattice Reduction Algorithm. Cryptology ePrint Archive, Report 2020/1237. `https://eprint.iacr.org/2020/1237`. 2020.

Idea: $(\alpha \cdot \mathrm{GH}(k_\alpha))^{\frac{1}{k_\alpha - 1}} \leq \mathrm{GH}(k)^{\frac{1}{k-1}}$



k_α is the smallest integer greater than k such that $\mathrm{GH}\,(k)^{\frac{1}{k-1}} \geq (\alpha \cdot \mathrm{GH}(k_\alpha))^{\frac{1}{k_\alpha - 1}}$ for $\alpha \geq 1$ and $k \geq 36$.

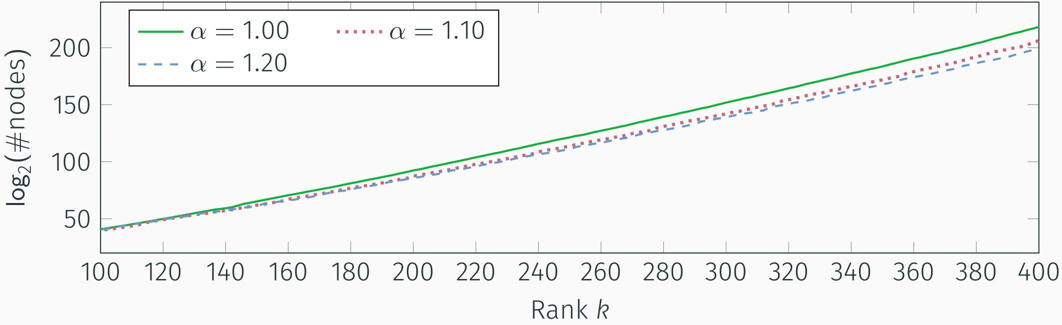
Theorem (Informal)

Let $\alpha > 1$ and $\rho \in (0, \frac{1}{2})$ be constants such that $4\,\alpha^4\,\rho \cdot (1-\rho) < 1$. Let $f(i)$ be a certain pruning function. Assume Heuristic 2 holds.

Let $T(n) := n^{c_0 n} \cdot 2^{c_1 n}$ be the cost of enumeration. Let k a sufficiently large integer. For any real $\eta \in [\frac{2\ln k}{\ln k - \ln(2\pi e^2)}, \frac{1}{2c_0})$, if $1 < \alpha \leq (k^{c_0} \cdot 2^{c_1})^2$, then some $(\alpha \cdot \mathrm{GH}(k_\alpha))$-HSVP enumeration oracle in rank k_α is exponentially faster than some $\mathrm{GH}(k)$-HSVP enumeration oracle in rank k by a multiplicative factor of at least

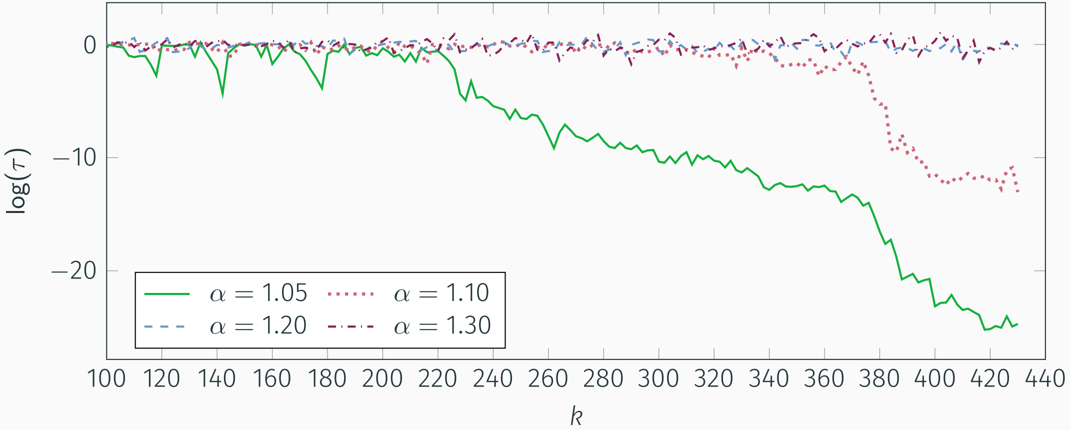
$$\alpha^{\left(\frac{1}{2} - c_0\eta\right)k} \cdot \left(4(1-\rho)\left(\frac{\sqrt{\alpha}}{(2e)^{c_0}\,2^{c_1}}\right)^{4\eta}\right)^{\frac{k\log\alpha}{4\log k}} \quad \text{(up to some polynomial factor).}$$

Practical Performance (Simulation)



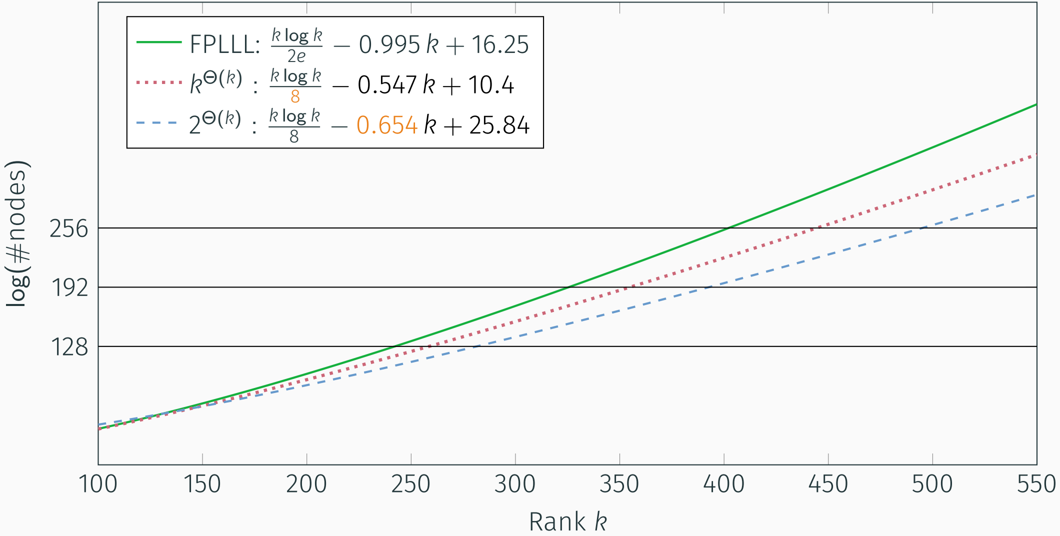
Expected cost $t_\alpha(k_\alpha)$ of the $(\alpha \cdot \mathbf{GH}(k_\alpha))$-HSVP enumeration oracle in rank k_α for reaching RHF $\mathbf{GH}\,(k)^{\frac{1}{k-1}}$.

Not-so-extreme Pruning



Expected number of solutions τ per enumeration for reaching RHF $\mathrm{GH}(k)^{\frac{1}{k-1}}$.

Thanks


FPLLL: $\frac{k \log k}{2e} - 0.995\,k + 16.25$
$k^{\Theta(k)}$: $\frac{k \log k}{8} - 0.547\,k + 10.4$
$2^{\Theta(k)}$: $\frac{k \log k}{8} - 0.654\,k + 25.84$
log(#nodes)
256
192
128
100
150
200
250
300
350
400
450
500
550
Rank k