LATTICE BASIS REDUCTION

6. LLL IMPROVEMENTS

Alfred Menezes
cryptography101.ca

November 2025

Outline

1. LLL recap
2. LLL refinements
3. Sieving and Enumeration
4. BKZ
5. Lattice challenges

LLL recap

- Let $B = [b_1, \ldots, b_n]$ be a basis for an integer lattice $L \subseteq \mathbb{Z}^n$.
- The LLL algorithm is a polynomial-time algorithm for finding an LLL-reduced basis for L.
- **Recall**: The Gram-Schmidt (GS) basis corresponding to B is $B^* = [b_1^*, \ldots, b_n^*]$, where $b_1^* = b_1$ and $b_i^* = \Pi_{i-1}(b_i)$ for $2 \leq i \leq n$, where $\Pi_{i-1}(b_i)$, the projection of b_i onto $\langle b_1^*, \ldots, b_{i-1}^* \rangle^{\perp}$, is defined as $\Pi_{i-1}(b_i) = b_i - \sum_{j<i} \mu_{ij} b_j^*$.

  The Gram-Schmidt coefficients are $\mu_{ij} = \langle b_i, b_j^* \rangle / \|b_j^*\|^2$ for $1 \leq j < i \leq n$.
- **Definition**. A lattice basis B is size-reduced if $|\mu_{ij}| \leq \frac{1}{2}$ for all $1 \leq j < i \leq n$.

  A size-reduced basis is LLL-reduced if the Lovász condition is satisfied:

  $$\|b_j^*\|^2 \geq (\tfrac{3}{4} - \mu_{j,j-1}^2) \, \|b_{j-1}^*\|^2 \text{ for all } 2 \leq j \leq n.$$
- **Theorem**: If $B = [b_1, \ldots, b_n]$ is LLL-reduced then $\|b_1\| \leq 2^{(n-1)/2} \lambda_1(L)$.

The LLL algorithm

Input: Lattice basis $B = [b_1, \ldots, b_n]$.

Output: LLL-reduced basis $[b_1, \ldots, b_n]$.

1. Size-reduce $[b_1, \ldots, b_n]$.
2. If there exists an index $j \in [2, n]$ such that $\|b_j^*\|^2 < (\frac{3}{4} - \mu_{j,j-1}^2)\,\|b_{j-1}^*\|^2$ then
   - Swap b_{j-1} and b_j.
   - Go to step 1.
3. Return($[b_1, \ldots, b_n]$).

Size reduction

Input: Lattice basis $B = [b_1, \ldots, b_n]$.

Output: Size-reduced basis $\tilde{B} = [\tilde{b}_1, \ldots, \tilde{b}_n]$.

1. Compute the GS basis $B^* = [b_1^*, \ldots, b_n^*]$.
2. For i from 2 to n do
   (a) For j from $i - 1$ to 1 do
   - Compute $\mu_{ij} = \langle b_i, b_j^* \rangle / \|b_j^*\|^2$.
   - Set $b_i \leftarrow b_i - \lfloor \mu_{ij} \rceil b_j$.
3. Return($\tilde{B} = [b_1, \ldots, b_n]$).

LLL refinement: the parameter δ

- The constant $\frac{3}{4}$ in the Lovász condition can be replaced by any constant δ, where $\frac{1}{4} < \delta < 1$.
- **Definition**. Let $\frac{1}{4} < \delta < 1$. A lattice basis B is $\delta-$LLL-reduced if it is size-reduced and
$$\|b_j^*\|^2 \geq (\delta - \mu_{j,j-1}^2)\,\|b_{j-1}^*\|^2 \text{ for all } 2 \leq j \leq n.$$
- **Theorem**. Let $\frac{1}{4} < \delta < 1$. If $B = [b_1, \ldots, b_n]$ is $\delta-$LLL-reduced, then
$$\|b_1\| \leq \left(\frac{4}{4\delta - 1}\right)^{(n-1)/2} \lambda_1(L).$$
- Choosing δ closer to 1 (e.g., $\delta = 0.99$) gives a smaller approximation factor, at the expense of a larger number of LLL iterations.
- Choosing $\delta = 1$ gives approximation factor $(2/\sqrt{3})^{n-1}$, but it not known whether LLL runs in polynomial time.

LLL refinement: Gram-Schmidt computations

- Instead of performing a full size-reduction after each swap, the Gram-Schimdt computations (B^* and μ_{ij}) can be scheduled so that only the necessary updates are computed.
- For example, after a Swap(b_9, b_{10}) operation, $b_1^*, \ldots, b_8^*$ remain unchanged, as do μ_{ij} for $1 \leq j < i \leq 8$.

Size reduction

Input: Lattice basis $B = [b_1, \ldots, b_n]$.

Output: Size-reduced basis $\tilde{B} = [\tilde{b}_1, \ldots, \tilde{b}_n]$.

1. Compute the GS basis $B^* = [b_1^*, \ldots, b_n^*]$.
2. For i from 2 to n do

   (a) For j from $i - 1$ to 1 do

   - Compute $\mu_{ij} = \langle b_i, b_j^* \rangle / \|b_j^*\|^2$.
   - Set $b_i \leftarrow b_i - \lfloor \mu_{ij} \rceil b_j$.
3. Return($\tilde{B} = [b_1, \ldots, b_n]$).

LLL refinement: deep insertion

- The Lovász condition $\|b_k^*\|^2 \geq (\frac{3}{4} - \mu_{k,k-1}^2)\,\|b_{k-1}^*\|^2$, which implies that $\|b_k^*\|^2 \geq \frac{1}{2}\|b_{k-1}^*\|^2$, guarantees that the GS basis vectors $b_i^{*\prime}$s do not get small too rapidly.
  - The Lovász condition can be written as: $\|\Pi_{k-2}(b_k)\|^2 \geq \frac{3}{4}\|b_{k-1}^*\|^2$.
  - If the Lovász condition is violated for some k, i.e., $\|\Pi_{k-2}(b_k)\|^2 < \frac{3}{4}\|b_{k-1}^*\|^2$, then b_{k-1} and b_k are swapped, so the new ordered basis is $(\ldots, b_{k-2}, b_k, b_{k-1}, b_{k+1}, \ldots)$.
- (Schnorr & Euchner, 1994) **Deep insertion**
  - Idea: Exchange non-adjacent basis vectors.
  - Insert b_k between b_{i-1} and b_i, where i is the smallest index such that $\|\Pi_{i-1}(b_k)\|^2 < \frac{3}{4}\|b_i^*\|^2$.
  - Note: Setting $i = k - 1$ gives the original Swap condition.
  - Experiments have shown that LLL with deep insertion yields shorter lattice bases, but at the expense of increased running time (in fact, the running time is not known to be polynomial-time).

LLL refinement: floating point

- The running time of LLL is $O(n^6 \log^3 X)$ bit operations, where $X = \max_i \|b_i\|$.
  - LLL performs arithmetic operations on integers of bitlength at most $O(n \log X)$.
- In practice, X can be large so the dependence on $\log^3 X$ is problematic.
  - Original LLL algorithm is only used in practice when n is small.
- Nguyen and Stehlé (2009): L^2 floating point variant of LLL.
  - The Gram-Schmidt computations (B^* and μ_{ij}) are performed using floating point arithmetic that requires a precision of only $n \log_2 3 \approx 1.58n$ bits.
  - Running time: $O(n^6 \log X + n^5 \log^2 X)$.

BKZ

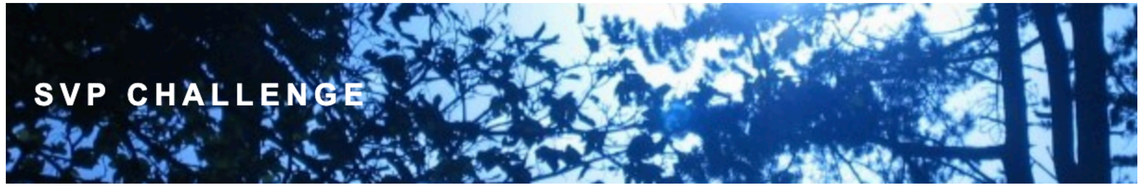
- Block Korkine-Zolotarev (BKZ): "block" generalization of LLL.
- Proposed by Schnorr and Euchner in 1994.
- Uses an exact SVP solver for lattices of relatively small dimensions $\leq \beta$, where β is the block size.
- Two algorithms for solving SVP exactly: enumeration and sieving.

Solving SVP exactly: Enumeration and Sieving

- Enumeration and sieving are two exponential-time strategies for solving SVP exactly.
  - **Enumeration**: $O(2^{0.18n \log n})$ time, modest storage.
  - **Sieving**: $O(2^{0.292n+o(n)})$ time, $O(2^{0.2075n+o(n)})$ storage.
- Sieving was introduced by Ajtai, Kumar and Sivakumar in 2001.
  - Sieving is faster than enumeration for dimension $n \geq 80$, but has exponential storage costs.
- For both enumeration and sieving, experimental run times are generally faster than what has been (heuristically) proven.

SVP challenges

- SVP instances in randomly-generated lattices of dimension n, where $n \leq 300$.
- The challenge is to find a lattice vector whose length is within a factor of 1.05 of the length of a shortest nonzero lattice vector as predicted by the Gaussian heuristic.
- All recent challenges solved used **sieving**.
- Z. Zhao, J. Ding and B.-Y.Yang report on their 2024 solution of the $n = 183$ SVP challenge.



https://www.latticechallenge.org/svp-challenge/

  - 112-core server, 0.87 Tbytes RAM, 30 calendar days.
  - "Sieving with streaming memory access", *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2025, pp. 362-384.
- Largest challenge solved as of October 2025: $n = 200$ by Ziyu Zhao and Jintai Ding.

LLL Swap operation

- $L \subseteq \mathbb{Z}^n$ is a full-rank lattice with basis $B = [b_1, \ldots, b_n]$.
- We have $\mathrm{vol}(L) = \prod_{i=1}^{n} \|b_i^*\|$, where $B^* = [b_1^*, \ldots, b_n^*]$.
- The LLL Swap operation is a local change, to ensure that the GS vectors do not shrink too quickly.
  - If $\|b_j^*\|^2 < (\frac{3}{4} - \mu_{j,j-1}^2)\|b_{j-1}^*\|^2$, then Swap$(b_{j-1}, b_j)$.
  - The new basis is $\tilde{B} = [\tilde{b}_1, \ldots, \tilde{b}_{j-1}, \tilde{b}_j, \ldots, \tilde{b}_n] = [b_1, \ldots, b_j, b_{j-1}, \ldots, b_n]$ with $\tilde{B}^* = [b_1^*, \ldots, b_{j-2}^*, \tilde{b}_{j-1}^*, \tilde{b}_j^*, b_{j+1}^*, \ldots, b_n^*]$.
  - We have $\|\tilde{b}_{j-1}^*\|^2 < \frac{3}{4}\|b_{j-1}^*\|^2$.
  - The Swap is followed by a size-reduction, which doesn't change the GS basis.
- Recall that $\|b_j^*\|^2 < (\frac{3}{4} - \mu_{j,j-1}^2)\|b_{j-1}^*\|^2$ can be restated as $\|\Pi_{j-2}(b_j)\|^2 < \frac{3}{4}\|\Pi_{j-2}(b_{j-1})\|^2$.

BKZ block operation

- Suppose that $B = [b_1, \ldots, b_n]$ is an LLL-reduced basis for L.
- For $i < j$, define $B[i,j] = [\Pi_{i-1}(b_i), \ldots, \Pi_{i-1}(b_j)]$, and let $L[i,j]$ be lattice spanned by $B[i,j]$.
  - $L[i,j]$ is the $(j-i+1)$ -dimensional sublattice formed by the projections of $b_i, \ldots, b_j$ onto $\langle b_1, \ldots, b_{i-1} \rangle^{\perp}$.
  - The first vector of $B[i,j]$ is b_i^*.
- The BKZ block operation with respect to $B[i,j]$ is the following:
  - Use an SVP solver to find $v' \in L[i,j]$, with $\|v'\| = \lambda_1(L[i,j])$.
  - Lift v' to a vector $v \in \langle b_1, b_2, \ldots, b_j \rangle$ with $\Pi_{i-1}(v) = v'$.
  - Let $B' = [b_1, \ldots, b_{i-1}, v, b_i, \ldots, b_j, b_{j+1}]$.
  - **Note**: The GS basis corresponding to B' is $[b_1^*, \ldots, b_{i-1}^*, v^*, \ldots]$, where $\|v^*\| = \|\Pi_{i-1}(v)\| = \|v'\| \leq \|\Pi_{i-1}(b_i)\| = \|b_i^*\|$.
  - Apply (modified) LLL to B' to remove the linear dependency and obtain $\tilde{B} = [\tilde{b}_1, \ldots, \tilde{b}_{j+1}]$.
  - The new basis for L is $B = [\tilde{b}_1, \ldots, \tilde{b}_{j+1}, b_{j+2}, \ldots, b_n]$.

BKZ algorithm

- Input: Basis $B = [b_1, \ldots, b_n]$ for a full-rank lattice L, block size β.
- Apply LLL to B.
- Repeatedly perform a BKZ tour:
  - Block operations on $B[1, \beta]$, $B[2, \beta+1]$, $B[3, \beta+2]$, … , $B[n-\beta+1, n]$.
  - Block operations on $B[n-\beta+2, n]$, $B[n-\beta+3, n]$, … , $B[n-1, n]$.
- Until the basis doesn't change.

BKZ analysis

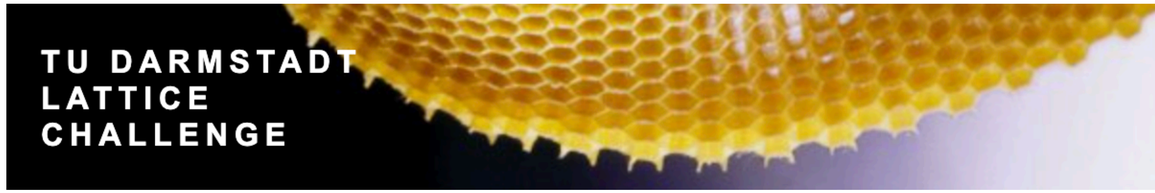
- As the block size β increases, the basis produced by BKZ becomes shorter. However, this comes at the expense of increased running time since the SVP solvers used run in exponential time (in β).
  - The overall BKZ running time depends on the cost of the SVP solver, and also the number of calls to the SVP solver.
  - There are guarantees on the quality of the basis produced by the BKZ algorithm.
- Roughly, the run time of BKZ is $O(2^\beta) \cdot \text{poly}(n)$, and the Hermite factor $\|b_1\|/\text{vol}(L)^{1/n}$ is $\beta^{O(n/\beta)}$.
  - **LLL** ($\beta = 2$): Polynomial running time, Hermite factor $\approx 2^{n/2}$.
  - **Sieving** ($\beta = n$): Exponential running time, Hermite factor $\approx \sqrt{n}$.
  - **BKZ** ($\beta = n^\alpha$): Subexponential running time $O(2^{n^\alpha})$, Hermite factor $2^{\tilde{O}(n^{1-\alpha})}$.
    - e.g., $\beta = \sqrt{n}$: Running time $O(2^{\sqrt{n}})$, Hermite factor $2^{\tilde{O}(\sqrt{n})}$.

BKZ in practice

- BKZ has not been proven to run in polynomial time.
  - In practice, BKZ is usually aborted before it terminates.
  - This makes sense since the quality of the basis increases more significantly in the earlier BKZ tours.
- Chen & Nguyen (Asiacrypt 2011) provided estimates for the required block size β to achieve a target Hermite factor $\|b_1\|/\text{vol}(L)^{1/n}$ with BKZ 2.0 (with early abort):
  - $\beta = 85, 106, 133, 168, 216, 286$ for target Hermite factor $1.01^n, 1.009^n, 1.008^n, 1.006^n, 1.005^n$, respectively.
  - See also "On the concrete hardness of Learning with Errors", M. Albrecht, R. Player and S. Scott, https://eprint.iacr.org/2015/46.

Lattice challenges

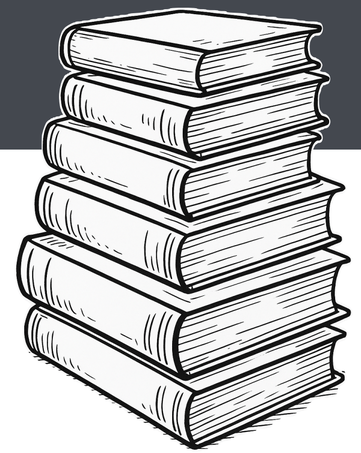
- SVP instances in specially chosen lattices of dimension n, where $n \leq 2000$.
- The challenge is to find a nonzero lattice vector that is shorter than the one listed.



https://www.latticechallenge.org

- Recent challenges solved used **BKZ+sieving**.
- Largest challenge solved as of October 2025: $n = 1100$ by Yao Sun.
  - August 15, 2024: found a lattice vector of length 175.40.
- Also: SVP, Ideal lattice, and LWE challenges.

Readings



1. P. Nguyen and D. Stehlé, "An LLL algorithm with quadratic complexity", *SIAM Journal on Computing*, 39 (2009), 874-903.

2. P. Nguyen and D. Stehlé, "LLL on the average", *Proceedings of ANTS VIII*, Lecture Notes in Computer Science, 4076 (2006), 238-256.

3. M. Yasuda, "A survey of solving SVP algorithms and recent strategies for solving the SVP challenge", *International Symposium on Mathematics, Quantum Theory, and Cryptography — Proceedings of MQC 2019*, 2021.

4. L. Ducas, L. Pulles and M. Stevens, "Towards a modern LLL implementation", https://eprint.iacr.org/2025/774.

5. J. Li and P. Nguyen, "A complete analysis of the BKZ lattice reduction algorithm", *Journal of Cryptology*, 2025.